

USER GUIDE

Essential Studio

for EJ2 Angular

Version - v25.2.3 | Release Date - May 08, 2024

Calendar	30
Getting started with Angular Calendar component	30
Dependencies.....	30
Setup Angular environment.....	30
Create a new application	30
Installing Syncfusion Calendar package	31
Registering Calendar module	31
Adding CSS reference.....	32
Adding Calendar component	32
Running the application.....	33
Setting the value, min and max dates.....	34
See Also	34
Date range in Angular Calendar component	35
Multi select in Angular Calendar component	36
Globalization in Angular Calendar component.....	36
Right-To-Left	40
Customization in Angular Calendar component.....	41
Disable Weekends.....	41
Day Cell Format.....	42
Highlight Weekends	46
See Also	46
Calendar views in Angular Calendar component.....	47
View Restriction	47
Accessibility in Angular Calendar component	49
WAI-ARIA attributes.....	50
Keyboard Interaction	50
Ensuring accessibility	52
See also	52
Islamic calendar in Angular Calendar component	52
Style appearance in Angular Calendar component	54
Customizing the background color for the Calendar	54
Customizing the Calendar date elements on hovering.....	54
Customizing the border of date cell grid	54
Customizing the Calendar title.....	54
Customizing the previous and next icon.....	55

Customizing the footer button	55
Customizing the selected date cell grid	55
Customizing the content header in Calendar	56
How To	56
Json data binding with calendar in Angular Calendar component	56
Two way binding in Angular Calendar component.....	57
Custom event emitter in Angular Calendar component.....	58
Set clear button in calendar in Angular Calendar component	59
Show dates of other months in Angular Calendar component	60
Select a sequence of dates in calendar in Angular Calendar component	61
Skip a month in calendar in Angular Calendar component	62
Customize the calendar day header in Angular Calendar component	63
Render the calendar with week numbers in Angular Calendar component	65
Change the first day of week in Angular Calendar component	65
Card.....	66
Getting started with Angular Card component	66
Dependencies.....	66
Setup Angular Environment.....	66
Create an Angular Application	67
Installing Syncfusion Card package	67
Adding CSS reference.....	68
Adding a simple Card	68
Adding a header to the card	68
See Also	69
Header content in Angular Card component.....	70
Header.....	70
Content	71
Card image in Angular Card component.....	72
Images	72
Divider	73
See Also	74
Action buttons in Angular Card component	74
Vertical	75
See Also	76
Horizontal in Angular Card component	76

Stacked cards	76
Style in Angular Card component	78
Customizing the card	78
Customizing the Header element	78
Customizing the card content.....	78
Divider used to separate the elements inside the card	78
Including image within card element	79
Including a title or caption for the image	79
To include heading image within the header	79
Customizing the Header main title	79
Customizing the Header subtitle.....	80
Including action buttons or anchor tags	80
To align card elements horizontally	80
To align elements vertically within the horizontal layout.....	80
How To	81
Customize the card image title position in Angular Card component.....	81
Integrate other component inside the card in Angular Card component	82
Carousel	83
Getting started with Angular Carousel component.....	83
Dependencies.....	83
Setup Angular environment.....	83
Create an Angular application	83
Installing Syncfusion Carousel Package.....	84
Adding Carousel module	84
Adding Syncfusion Carousel component	85
Adding CSS reference.....	87
Running the application	87
Populating items in Angular Carousel component	89
Populating items using carousel item	89
Populating items using data source	90
Selection.....	91
Partial visible slides	95
See Also	98
Navigators and indicators in Angular Carousel component	98
Navigators	98

Indicators	103
Play button	114
Animations and transitions in Angular Carousel component	117
Animations	117
Intervals between slides	120
Auto play slides	122
Pause on hover.....	124
Looping slides.....	125
Slide changing events.....	127
Disable touch swiping	129
Swipe Modes.....	130
Accessibility in Angular Carousel component.....	132
ARIA attributes.....	133
Keyboard interaction	133
Ensuring accessibility	134
See also	134
Chart.....	134
Getting started with Angular Chart component.....	134
Setup Angular Environment.....	134
Create an Angular Application	134
Installing Syncfusion Chart package.....	135
Registering Chart Module	135
Module Injection.....	137
Populate Chart with Data.....	138
Add Chart Title	140
Enable Legend.....	142
Add Data Label	143
Enable Tooltip	144
Working with data in Angular Chart component.....	145
Local Data.....	146
Lazy loading.....	148
Remote Data	150
Empty points	151
Chart dimensions in Angular Chart component	154
Size for Container.....	154

Size for Chart	156
Category axis in Angular Chart component	158
Labels Placement	159
Range	160
Indexed category axis.....	161
Numeric axis in Angular Chart component.....	162
Range	163
Range Padding.....	164
Label Format	169
GroupingSeparator	171
Custom Label Format.....	172
Date time axis in Angular Chart component.....	173
DateTime Axis	173
DateTimeCategory Axis.....	174
Label Format	180
Logarithmic axis in Angular Chart component.....	182
Range	183
Logarithmic Base.....	184
Logarithmic Interval	185
Axis labels in Angular Chart component	186
Smart Axis Labels	186
Axis Labels Positioning	189
Multilevel Labels	190
Edge Label Placement	196
Trim using maximum label width.....	197
Labels Customization	198
Customizing Specific Point	201
Line break support	202
Maximum Labels	203
Axis customization in Angular Chart component.....	205
Axis Crossing	205
Title	206
Title Rotation.....	207
Tick Lines Customization	208
Grid Lines Customization	209

Multiple Axis	211
Inversed Axis	212
Opposed Position	213
Strip line in Angular Chart component	214
Horizontal Strip lines.....	214
Vertical Striplines	215
Customize the strip line	216
Customize the stripline text.....	218
Dash Array.....	219
Recurrence Stripline.....	220
Size Type	221
Segment Stripline.....	222
See Also	224
Multiple panes in Angular Chart component.....	224
Rows.....	224
Columns	227
Chart Types	230
Line Chart in Angular Chart component	230
Step line chart in Angular Chart component	234
Stacked Line Chart in Angular Chart component.....	236
100% Stacked Line Chart in Angular Chart component.....	239
Spline Chart in Angular Chart component	242
Area Chart in Angular Chart component	244
Range Area in Angular Chart component	249
Range step area in Angular Chart component.....	252
Spline Range Area in Angular Chart component.....	255
Stacked Area in Angular Chart component.....	258
100% Stacked Area in Angular Chart component.....	260
Stacked step area in Angular Chart component	263
Step area in Angular Chart component	265
Spline Area in Angular Chart component	268
Column in Angular Chart component	270
Range Column in Angular Chart component	277
Stacked Column in Angular Chart component.....	279
100% Stacked Column in Angular Chart component.....	285

Bar in Angular Chart component	289
Stacked Bar in Angular Chart component.....	294
100% Stacked Bar in Angular Chart component.....	297
Scatter in Angular Chart component	300
Bubble in Angular Chart component	303
Polar in Angular Chart component	306
Radar in Angular Chart component	317
Hilo in Angular Chart component	320
High Low Open Close in Angular Chart component.....	323
Candle in Angular Chart component.....	325
Box and Whisker in Angular Chart component.....	328
Waterfall in Angular Chart component.....	331
Histogram in Angular Chart component	334
Error Bar in Angular Chart component	336
Vertical Chart in Angular Chart component.....	343
Pareto in Angular Chart component	344
Chart series in Angular Chart component.....	347
Multiple Series	347
Combination Series	349
Enable Complex Property in Series	350
Technical indicators in Angular Chart component.....	351
Accumulation Distribution	351
Average True Range (ATR)	355
Bollinger Band	358
Exponential Moving Average (EMA)	365
Momentum	368
Moving Average Convergence Divergence (MACD)	375
Relative Strength Index (RSI).....	382
Simple Moving Average (SMA)	385
Stochastic	388
Triangular Moving Average (TMA)	395
Trend lines in Angular Chart component.....	405
Linear.....	405
Exponential	407
Logarithmic	408

Polynomial	410
Power	411
Moving Average	412
Forecasting.....	415
Show or hide a trendline.....	418
Data markers in Angular Chart component.....	419
Marker.....	420
Shape.....	421
Images	422
Customization	423
Customizing specific point	423
Fill marker with series color	425
See also	426
Data labels in Angular Chart component.....	426
Position	427
Data Label Template	428
Format.....	429
Text Mapping	430
Margin.....	431
DataLabel Rotation	432
Customization	433
Customizing Specific Point	435
Show percentage based on each series points.....	436
See Also	438
Chart annotations in Angular Chart component.....	438
Region	440
Co-ordinate Units.....	441
Alignment.....	442
Adding y-axis sub title through on annotation	443
See Also	445
Legend in Angular Chart component	445
Position and Alignment.....	445
Customization	450
Set the label color based on series color	457
Series Selection on Legend	458

Enable Animation	460
Collapsing Legend Item	461
Legend Title	462
Arrow Page Navigation	464
Legend Item Padding	465
See Also	466
Tooltip in Angular Chart component	467
Default tooltip	467
Fixed tooltip	468
Format the tooltip	469
Individual series format	470
Tooltip template	471
Tooltip mapping name	472
Customize the appearance of tooltip	473
See also	475
Zooming in Angular Chart component	475
Enable zooming	475
Modes	476
Toolbar	478
Enable pan	479
Enable scrollbar	480
Auto interval on zooming	481
Data editing in Angular Chart component	483
Enable Data Editing	483
Cross hair and track ball in Angular Chart component	484
Tooltip for axis	486
Customization	487
Trackball	488
Synchronized Charts in Angular Chart component	490
Tooltip synchronization	490
Crosshair synchronization	493
Zooming synchronization	496
Selection synchronization	499
Selection in Angular Chart component	502
Point	502

Series.....	504
Cluster	505
Rectangular selection.....	507
Lasso selection	508
Multi-region selection.....	510
Selection type.....	511
Selection on load.....	513
Selection through on legend.....	514
Customization for selection	516
See Also	518
Chart print in Angular Chart component	518
Print.....	518
Export.....	519
Multiple Chart Export.....	525
Exporting chart using base64 string.....	526
Chart appearance in Angular Chart component.....	528
Custom color palette.....	528
Data point customization.....	530
Point level customization.....	533
Chart area customization.....	534
Animation.....	538
Fluid animation	540
Chart title	542
Chart subTitle	549
See also	550
Render methods in Angular Chart component.....	550
SVG	551
Canvas	551
Accessibility in Angular Chart component	551
Keyboard interaction	552
Ensuring accessibility	553
See also	553
Internationalization in Angular Chart component.....	553
Localization in Angular Chart component.....	555
Ej1 api migration in Angular Chart component	557

Chart.....	557
3DChart	559
Annotations.....	559
Columns	560
Rows.....	561
Common Series Options	562
Crosshair	562
Indicator	562
Legend.....	566
PrimaryXAxis	568
PrimaryYAxis	576
Axes.....	584
Series.....	591
Marker.....	597
Error bar	600
Trendlines.....	602
Striplines	604
Multilevel labels.....	606
Methods.....	607
Events.....	608
How To	612
Live chart in Angular Chart component	612
Prevent data label in Angular Chart component	615
Tool tip format in Angular Chart component	616
Add series in Angular Chart component	618
Points customization in Angular Chart component	620
Stacking total in Angular Chart component.....	621
Selected data grid in Angular Chart component.....	623
Marker customization in Angular Chart component	625
Legend customization in Angular Chart component	627
Tool tip table in Angular Chart component	629
Footer in Angular Chart component	630
watermark for chart.....	631
footer for chart	631
Threshold in Angular Chart component.....	633

Grid data chart in Angular Chart component	634
Data label template in Angular Chart component.....	636
Hide tool tip in Angular Chart component.....	638
Dotted line in Angular Chart component.....	640
Initial scrollbar in Angular Chart component.....	642
Dialog chart in Angular Chart component	644
Series visible in Angular Chart component	646
Dynamic chart in Angular Chart component	648
Database data in Angular Chart component	650
Column width in Angular Chart component	654
Customize scatter chart in Angular Chart component	656
Overlap area in Angular Chart component	661
CheckBox.....	663
Getting started with Angular Check box component	663
Dependencies.....	663
Setup Angular environment.....	663
Create an Angular application	664
Installing Syncfusion CheckBox package.....	664
Adding CheckBox module	665
Adding Syncfusion CheckBox component.....	665
Adding CSS reference.....	665
Running the application.....	666
Change the CheckBox state	666
Label and size in Angular Check box component.....	667
Label.....	667
Size	668
See Also	669
Accessibility in Angular Check box component	669
WAI-ARIA attributes.....	670
Keyboard interaction	670
Ensuring accessibility	670
See also	670
How To	670
Bind data using two way binding in Angular Check box component.....	670
Customized checkbox in Angular Check box component	672

Name and value in form submit in Angular Check box component	679
Right to left in Angular Check box component	680
Ej1 api migration in Angular Check box component.....	680
Properties.....	680
Methods	681
Events.....	682
Chips.....	682
Getting started with Angular Chips component	682
Setup Angular Environment.....	683
Create an Angular Application	683
Installing Syncfusion Chips package.....	683
Registering ChipList Module	684
Adding CSS reference.....	684
Add Chip.....	684
Run the application	685
Types in Angular Chips component	685
Input Chip.....	686
Choice Chip	686
Filter Chip	687
Action Chip	688
Customization in Angular Chips component.....	689
Styles	689
Leading Icon	690
Avatar.....	691
Avatar Content.....	692
Trailing Icon.....	692
Outline Chip	693
Style in Angular Chips component.....	694
Customizing the chip text	694
Customizing the chip icon	694
Customizing the chip delete button.....	694
Customizing the chip outline	695
Customizing the chip on selection	695
Customizing the chip avatar text	695
Accessibility in Angular Chips component	696

WAI-ARIA attributes	697
Keyboard interaction	697
Ensuring accessibility	697
See also	697
3D Circular Chart	697
Getting started with Angular 3D Circular Chart component	697
Setup angular environment	697
Create an Angular application	698
Installing Syncfusion 3D Circular Chart package	698
Registering 3D Circular Chart module	699
Pie and donut in Angular 3D Circular Chart component	701
Pie chart	701
Radius customization	702
Various radius pie chart	703
Donut chart	704
Text and fill color mapping	705
Customization	706
Data Label in Angular 3D Circular Chart component	707
Positioning	708
Data label template	709
Connector line	710
Text mapping	711
Format	712
Customization	713
Using textRender event	715
Using template	716
Empty points in Angular 3D Circular Chart component	717
Customization	718
Legend in Angular 3D Circular Chart component	719
Position and alignment	720
Legend reverse	721
Legend shape	722
Legend size	723
Legend item size	724
Legend paging	725

Legend text wrap	726
Legend title	727
Arrow page navigation	728
Legend item padding	729
Tooltip in Angular 3D Circular Chart component.....	730
Header.....	731
Format.....	732
Tooltip template	733
Fixed tooltip	734
Customization	735
Customization of individual tooltip.....	736
Title	737
Title customization.....	738
Subtitle	739
Subtitle customization	741
Print and Export in Angular 3D Circular Chart component.....	742
Print.....	742
Export.....	743
Circular Gauge.....	744
Getting started with Angular Circular gauge component.....	744
Setup Angular Environment.....	744
Create an Angular Application	744
Installing Syncfusion Circular Gauge package.....	745
Registering Circular Gauge Module	745
Set Pointer Value	747
Gauge dimensions in Angular Circular gauge component.....	748
Size for Container.....	748
Size for Circular Gauge	749
Gauge axes in Angular Circular gauge component	750
Axis Customization.....	750
Angles and Direction	751
Axis Radius	752
Ticks.....	754
Labels	756
Minimum and Maximum	762

Multiple Axes	763
Gauge ranges in Angular Circular gauge component	764
Start and End.....	764
Customization	764
Radius.....	765
Dragging Range	767
Multiple Ranges	768
Rounded corner radius	769
Gradient Color.....	770
See also	776
Gauge pointers in Angular Circular Gauge Component.....	776
Needle Pointers.....	776
RangeBar Pointer	780
Rounded corner for range bar pointer	782
Marker Pointer	783
Dragging Pointer	785
Multiple Pointers	786
Animation.....	787
Gradient Color.....	788
Gauge annotations in Angular Circular gauge component	792
Content	792
Position	793
Sub Gauge	794
See also	797
Animation in Angular Circular Gauge component	797
Gauge legend in Angular Circular gauge component	799
Legend customization	799
Position and alignment	799
Font of the legend text	800
Toggle option in legend	802
Paging support in legend	803
Legend text customization.....	804
Gauge user interaction in Angular Circular gauge component.....	806
Tooltip for pointers	806
Tooltip for ranges.....	808

Tooltip for annotations	808
Pointer Drag	810
Gauge print and export in Angular Circular gauge component	811
Print.....	811
Export.....	811
Gauge appearance in Angular Circular gauge component	814
Gauge Title	814
Gauge Position	815
Area Customization.....	817
Radius calculation based on angles	819
Accessibility in Angular Circular gauge component.....	820
WAI-ARIA attributes.....	820
Screen reading in Circular Gauge.....	821
Ensuring accessibility	821
See also	821
Internationalization in Angular Circular Gauge component	821
Globalization	821
Right-to-left.....	822
Ej1 api migration in Angular Circular gauge component	824
Circular gauge dimensions	824
Axis Line	825
Ticks.....	826
Labels	827
Ranges.....	829
Needle Pointer	830
Marker Pointer.....	831
Rangebar Pointer	832
Annotations.....	832
Appearance	833
Events.....	834
ColorPicker.....	836
Getting started with Angular Color picker component	836
Dependencies.....	836
Setup Angular environment.....	836
Create an Angular application	836

Installing Syncfusion ColorPicker package	837
Adding ColorPicker module	837
Adding Syncfusion ColorPicker component.....	838
Adding CSS reference.....	838
Running the application	839
Inline type	839
Mode and value in Angular Color picker component.....	840
Rendering palette at initial load	840
Color value	841
See Also	841
Localization in Angular Color picker component	842
Localization	842
Right to Left - RTL.....	843
See Also	843
Accessibility in Angular Color picker component.....	844
WAI-ARIA attributes.....	845
Keyboard interaction	845
Ensuring accessibility	845
See also	845
Style and appearance in Angular Color picker component	846
How To	846
Hide control buttons in Angular Color picker component.....	846
Render palette alone in Angular Color picker component	847
Two way binding in Angular Color picker component.....	847
Colorpicker in dropdownbutton in Angular Color picker component	849
Customize colorpicker in Angular Color picker component	850
Handle no color support in Angular Color picker component	857
Disabled in Angular Color picker component	860
Ej1 api migration in Angular Color picker component.....	860
Properties.....	860
Methods	862
Events.....	864
ComboBox.....	864
Getting started with Angular Combo box component	864
Dependencies.....	865

Setup angular environment	865
Create a new application	865
Installing Syncfusion ComboBox package	866
Registering ComboBox module.....	866
Adding CSS reference.....	867
Adding ComboBox component	867
Binding data source	868
Running the application	868
Custom values.....	869
Configure the popup list	870
Two-way binding.....	871
See Also	872
Data binding in Angular Combo box component.....	872
Binding local data.....	872
Binding remote data	875
Data binding using Async pipe	876
See Also	878
Value binding in ComboBox Component.....	878
Primitive Data Types	878
Object Data Types	879
Templates in Angular Combo box component	880
Item template	880
Group template.....	881
Header template	882
Footer template	884
No records template	885
Action failure template	886
See Also	887
Virtualization in ComboBox Component	887
Binding local data.....	887
Binding remote data	888
Grouping	890
Filtering with Virtualization.....	891
Grouping in Angular Combo box component	892
Customization	893

See Also	893
Filtering in Angular Combo box component	894
Limit the minimum filter character	895
Change the filter type	896
Case sensitive filtering	897
Diacritics Filtering.....	899
See Also	900
Localization in Angular Combo box component	900
Loading translations.....	900
See Also	901
Style in Angular Combo box component	901
Customizing the appearance of wrapper element	901
Customizing the dropdown icon's color	902
Customizing the focus color.....	902
Customizing the outline theme's focus color	902
Customizing the disabled component's text color	902
Customizing the float label element's focusing color	903
Customizing the color of the placeholder text	903
Customizing the text selection color.....	903
Customizing the background color of focus, hover, and active item's	903
Customizing the appearance of pop-up element	904
Adding mandatory asterisk to placeholder and float label.....	904
Accessibility in Angular Combo box component	905
WAI-ARIA attributes.....	906
Keyboard interaction	906
Ensuring accessibility	908
See also	908
Form support in Angular Combo box component	908
Template-Driven Forms	908
Reactive Forms.....	909
How To	910
Autofill in Angular Combo box component	910
Cascading in Angular Combo box component.....	911
Icons support in Angular Combo box component	913
Ej1 api migration in Angular Combo box component.....	914

DataBinding.....	914
Filtering	915
Template	915
Applying CSS.....	916
Grouping	916
Accessibility.....	916
Placeholder	917
Miscellaneous	917
Sorting.....	917
Selection.....	917
Popup.....	918
Common.....	918
ContextMenu	919
Getting started with Angular Context menu component	919
Dependencies.....	919
Setup Angular environment.....	920
Create an Angular application	920
Installing Syncfusion ContextMenu Package	920
Adding ContextMenu module.....	921
Adding Syncfusion ContextMenu component	921
Adding CSS reference.....	922
target {	922
Running the application.....	923
Rendering items with Separator	924
Icons and navigation in Angular Context menu component	925
Icons	925
Navigation	926
See Also	927
Templates in Angular Context menu component.....	927
Template	927
Multilevel nesting	928
See Also	929
Accessibility in Angular Context menu component.....	929
WAI-ARIA attributes.....	930
Keyboard interaction	931

Ensuring accessibility	931
See also	931
Style and appearance in Angular Context menu component.....	931
How To	931
Populate menu items with data source in Angular Context menu component	931
Context menu item click in Angular Context menu component	933
Open and close contextmenu in Angular Context menu component	934
Change menu items dynamically in Angular Context menu component	935
Template in Angular Context menu component	936
Underline a character in the item text in Angular Context menu component.....	940
Open a dialog on contextmenu item click in Angular Context menu component	940
Change animation settings in Angular Context menu component.....	942
Add or remove context menu items in Angular Context menu component	943
Enable or disable context menu items in Angular Context menu component.....	945
Dashboard Layout	947
Getting started with Angular Dashboard layout component	947
Prerequisites	947
Setting up angular project	947
Adding Dependencies	947
Installing Syncfusion DashboardLayout package	948
Adding style sheet to the application	948
Add DashboardLayout to the application	949
Setting the `panels` property using HTML attributes	949
Run the application	951
Setting the `panels` property through binding	953
Setting size of cells in Angular Dashboard layout component	955
Modifying cell size.....	955
Setting cell spacing.....	956
Graphical representation of layout.....	957
Rendering component in right-to-left direction	958
Panels	959
Position sizing of panels in Angular Dashboard layout component	959
Setting header of panels in Angular Dashboard layout component.....	962
Add remove panels in Angular Dashboard layout component.....	966
Interaction With Panels	969

Dragging moving of panels in Angular Dashboard layout component	969
Moving panels in Angular Dashboard layout component	974
Resizing of panels in Angular Dashboard layout component	976
Floating of panels in Angular Dashboard layout component	979
Responsive adaptive in Angular Dashboard layout component.....	980
Save restore in Angular Dashboard layout component.....	981
Style in Angular Dashboard layout component	983
Customizing the dashboard layout panel header	983
Customizing the dashboard layout panel content.....	983
Customizing the dashboard layout panel resize icon	983
Customizing the dashboard layout panel background	984
Accessibility in Angular Dashboard Layout component	984
WAI-ARIA attributes.....	985
Keyboard interaction	985
Ensuring accessibility	985
See also	985
How To	986
Initializing dashboard using systemjs in Angular Dashboard layout component	986
DataManager	992
Getting started with Angular Data component	992
Dependencies.....	992
Setup Angular Environment.....	993
Create an Angular Application	993
Installing Syncfusion Data package.....	993
Connection to a data source	994
Filter	999
Sort.....	1000
Page.....	1002
Component binding	1003
Data binding in Angular Data component	1004
Local data binding	1005
Remote data binding.....	1006
See Also	1007
Adaptors in Angular Data component	1007
Json adaptor.....	1007

Url adaptor	1009
OData adaptor	1009
ODataV4 adaptor	1010
Web API adaptor	1012
WebMethod Adaptor.....	1012
GraphQL Adaptor	1013
Writing custom adaptor.....	1018
Querying in Angular Data component	1019
Specifying resource name using `from`	1019
Projection using `select`	1021
Eager loading navigation properties	1022
Sorting	1023
Filtering	1024
Searching.....	1027
Grouping	1028
Paging.....	1030
Aggregation.....	1031
Hierarchical query	1032
Manipulation in Angular Data component	1034
Insert	1034
Update.....	1036
Remove	1038
Batch Edit Operation.....	1040
How to in Angular Data component	1042
Work in offline mode	1042
Sending additional parameters to server	1044
Adding custom headers	1045
DatePicker	1045
Getting started with Angular Datepicker component	1045
Dependencies.....	1045
Setup Angular environment.....	1046
Create a new application	1046
Installing Syncfusion DatePicker package	1046
Registering DatePicker module.....	1047
Adding CSS reference.....	1048

Adding DatePicker component	1048
Running the application	1048
Setting the selected date	1049
Setting the date range to restrict selection	1050
See Also	1050
Date range in Angular Datepicker component	1051
Globalization in Angular Datepicker component.....	1052
Right-To-Left	1054
Date format in Angular Datepicker component	1055
Date Format	1055
Parse and Format Date value based on culture-specific formats.....	1056
Date masking in Angular Datepicker component	1057
Configure Mask Placeholder	1059
Strict mode in Angular Datepicker component	1060
Customization in Angular Datepicker component.....	1062
Adding mandatory asterisk to placeholder and float label.....	1063
See Also	1064
Date views in Angular Datepicker component	1064
Start view	1064
Depth view restriction	1065
Accessibility in Angular Datepicker component	1065
WAI-ARIA attributes.....	1066
Keyboard Interaction	1067
Input Element.....	1067
Calendar Navigation.....	1067
Ensuring accessibility	1068
See also	1068
Style appearance in Angular Datepicker component	1068
Customizing the appearance of DatePicker wrapper element.....	1068
Customizing the DatePicker icon element.....	1069
Customizing the Calendar popup of the DatePicker.....	1069
Full screen mode support in mobiles and tablets.....	1069
How To	1071
Json data binding in Angular Datepicker component.....	1071
Two way binding in Angular Datepicker component.....	1072

Disable placeholder readonly in Angular Datepicker component	1073
Prevent the popup close in Angular Datepicker component.....	1073
Open datepicker popup on input click in Angular Datepicker component	1074
Css customization in Angular Datepicker component	1075
Custom event emitter in Angular Datepicker component.....	1076
Custom validation using form validator in Angular Datepicker component	1077
Reactive form in Angular Datepicker component	1078
Template driven forms in Angular Datepicker component	1080
Customize the datepicker day header in Angular Datepicker component.....	1082
Ej1 api migration in Angular Datepicker component.....	1083
Date Selection	1083
Date Format	1083
Calendar Views.....	1084
Date Range	1084
Disabled Dates	1084
Customization	1084
Accessibility	1087
Persistence	1087
Validation	1087
Common.....	1088
Globalization	1089
Strict Mode	1090
Open and Close	1090
View Navigation	1090
DateRangePicker	1091
Getting started with Angular Daterangepicker component	1091
Dependencies.....	1091
Setup Angular environment.....	1091
Create a new application	1091
Installing Syncfusion DateRangePicker package	1092
Registering DateRangePicker module.....	1092
Adding CSS reference.....	1093
Adding DateRangePicker component	1093
Running the application	1094
Setting the start and end date	1094

See Also	1095
Range selection in Angular Daterangepicker component	1095
Restrict the range within a range.....	1095
Range span.....	1097
Strict mode.....	1097
Globalization in Angular Daterangepicker component	1098
Right-To-Left	1102
Date format customization	1104
Customization in Angular Daterangepicker component.....	1105
Day cell format.....	1105
First day of week	1106
Preset Ranges.....	1107
See Also	1108
Accessibility in Angular Daterangepicker component	1108
WAI-ARIA attributes.....	1109
Keyboard Interaction	1109
Input Navigation.....	1109
Calendar Navigation.....	1110
Ensuring accessibility	1111
See also	1111
Style appearance in Angular Daterangepicker component.....	1111
Customizing the appearance of DateRangePicker wrapper element.....	1111
Customizing the DateRangePicker icon element.....	1112
Customizing the DateRangePicker popup calendar header	1112
Customizing the DateRangePicker popup calendar header title	1112
Customizing the DateRangePicker popup calendar content	1112
Customizing the DateRangePicker popup calendar content title.....	1113
Customizing the DateRangePicker popup calendar previous and next icon	1113
Customizing the DateRangePicker popup calendar date cell grid on hovering.....	1113
Customizing the DateRangePicker popup calendar primary button in footer	1113
Customizing the DateRangePicker popup calendar cancel button in footer.....	1114
Customizing the footer element in the DateRangePicker popup calendar	1114
Customizing the selected date cell grid in the DateRangePicker popup calendar	1114
Full screen mode support in mobiles and tablets.....	1115
How To	1117

Two way binding in Angular Daterangepicker component	1117
Disable placeholder readonly in Angular Daterangepicker component.....	1117
Customization using cssclass in Angular Daterangepicker component.....	1118
Custom event emitter in Angular Daterangepicker component	1120
Custom validation using form validator in Angular Daterangepicker component	1121
Reactive form in Angular Daterangepicker component	1122
Template driven forms in Angular Daterangepicker component.....	1123
Customize the daterangepicker day header in Angular Daterangepicker component	1125
Ej1 api migration in Angular Daterangepicker component	1127
Date Selection	1127
Date Format	1127
Date Range	1127
Disabled Dates	1129
Customization	1129
Accessibility.....	1131
Persistence	1131
Common.....	1131
Globalization	1132
Strict mode.....	1133
Open and Close	1133

Calendar

Getting started with Angular Calendar component

The following section explains the steps required to create a simple Calendar component and also it demonstrates the basic usage of the Calendar. To get started quickly with angular DropDownList component using angular CLI, you can check the video below.

Dependencies

Install the below required dependency package in order to use the **Calendar** component in your application.

```
`javascript
|-- @syncfusion/ej2-angular-calendars
|-- @syncfusion/ej2-angular-base
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-calendars
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-splitbuttons
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-buttons
`,`
```

Setup Angular environment

Angular provides the easiest way to set angular CLI projects using [Angular CLI](#) tool.

Install the CLI application globally to your machine.

```
`bash
npm install -g @angular/cli
`,`
```

Create a new application

```
`bash
ng new syncfusion-angular-calendar
`,`
```

By default, it install the CSS style base application. To setup with SCSS, pass `--style=scss` argument on create project.

Example code snippet.

```
`bash
ng new syncfusion-angular-calendar --style=scss
`,`
```

Navigate to the created project folder.

```
`bash
```

```
cd syncfusion-angular-calendar
```

```
,
```

Installing Syncfusion Calendar package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(`>=20.2.36`) has been moved to the Ivy distribution to support the Angular [ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-calendars](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-calendars --save
```

```
,
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-calendars@ngcc](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-calendars@ngcc --save
```

```
,
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
```

```
@syncfusion/ej2-angular-calendars:"20.2.38-ngcc"
```

```
,
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Registering Calendar module

Import Calendar module into Angular application(`src/app/app.module.ts`) from the package `@syncfusion/ej2-angular-calendars`.

```
`javascript
```

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// import the CalendarModule for the Calendar component
import { CalendarModule } from '@syncfusion/ej2-angular-calendars';
import { AppComponent } from './app.component';

@NgModule({
  //declaration of CalendarModule into NgModule
  imports: [ BrowserModule, CalendarModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
`
```

Adding CSS reference

The following CSS files are available in `../node_modules/@syncfusion` package folder.

This can be referenced in [src/styles.css] using following code.

```
`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-angular-calendars/styles/material.css';
`
```

If you want to refer the combined component styles, please make use of our [CRG](#) (Custom Resource Generator) in your application.

Adding Calendar component

Modify the template in [src/app/app.component.ts] file to render the Calendar component. by using `<ejs-calendar>` selector.

```
`javascript
import { Component } from '@angular/core';

@Component({
```



```

selector: 'app-root',
template: `<!-- To Render Calendar -->
<ejs-calendar></ejs-calendar>`
})
export class AppComponent { }
`

```

Running the application

After completing the configuration required to render a basic Calendar, run the following command to display the output in your default browser.

```

ng serve
`

```

The following example illustrates the output in your browser

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [

    CalendarModule //declaration of ej2-angular-calendars module into
NgModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <!-- To Render Calendar -->
    <ejs-calendar></ejs-calendar>`
})
export class AppComponent {
  constructor() {
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Setting the value, min and max dates

The following example demonstrates how to set the value, min and max dates on initializing the Calendar. Here the Calendar allows you to select a date within a range from 9th to 15th. To know more about range restriction in Calendar, please refer this [page](#).

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [

    CalendarModule //declaration of ej2-angular-calendars module into
    NgModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <!-- Sets the value, min and max -->
    <ejs-calendar [value]='dateValue' [min]='minDate'
    [max]='maxDate'></ejs-calendar>
  `
})
export class AppComponent {
  public month: number = new Date().getMonth();
  public fullYear: number = new Date().getFullYear();
  public dateValue: Date = new Date(this.fullYear, this.month, 11);
  public minDate: Date = new Date(this.fullYear, this.month, 9);
  public maxDate: Date = new Date(this.fullYear, this.month, 15);
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

You can refer to our [Angular Calendar](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Calendar example](#) that shows how to render the Calendar in Angular.

See Also

- [Select multiple dates in the Calendar](#)
- [Render Calendar with specific culture](#)
- [How to change the initial view of the Calendar](#)
- [Render Calendar with week numbers](#)
- [Show other month dates](#)

Date range in Angular Calendar component

You can restrict the user to select the date from a specified range of dates by utilizing the [min](#) and [max](#) properties. Always the min date has to be lesser than the max date.

The below example allows you to select a date within a range from 7th to 27th days in a month.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [

    CalendarModule //declaration of ej2-angular-calendars module into
    NgModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <!--Sets the value, min and max -->
    <ejs-calendar [value]='dateValue' [min]='minDate'
    [max]='maxDate'></ejs-calendar>
  `
})
export class AppComponent {
  public today: Date = new Date();
  public currentYear: number = this.today.getFullYear();
  public currentMonth: number = this.today.getMonth();
  public currentDay: number = this.today.getDate();
  public dateValue: Object = new Date(new Date().setDate(14));
  public minDate: Object = new Date(this.currentYear, this.currentMonth,
7);
  public maxDate: Object = new Date(this.currentYear, this.currentMonth,
27);
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

If the value of [min](#) or [max](#) properties changed through code behind, then you have to update the [value](#) property to set within the range. Or else, if the value is out of specified date range and less than [min](#) date, value property will be updated with min date or the value is higher than max date, value property will be updated with [max](#) date.

Multi select in Angular Calendar component

Calendar provides an option to select **single** or **multiple dates** or **sequence of dates** by using [isMultiSelection](#) and [values](#) properties. By default, [isMultiSelection](#) property will be in disabled state.

The following example demonstrates the functionality of [isMultiSelection](#) property and [values](#) properties in the Calendar control.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [

    CalendarModule //declaration of ej2-angular-calendars module into
    NgModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <!-- Sets the isMultiSelection and values properties-->
    <ejs-calendar [values]='dateValues'
    [isMultiSelection]='multiSelect'></ejs-calendar>
  `
})
export class AppComponent {
  public dateValues: Date[] = [new Date('1/1/2020'), new
  Date('1/15/2020'), new Date('1/3/2020'), new Date('1/25/2020')];
  public multiSelect: Boolean = true;
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Globalization in Angular Calendar component

Globalization is the combination of internalization and localization. You can adapt the component to various languages by parsing and formatting the date or number [Internationalization](#) and also add culture specific customization and translation to the text [localization](#).

By default, Calendar date format, week and month names are specific to American English culture. It utilizes the [Essential JavaScript 2 Internationalization](#) package to parse and format the date object based on the culture by uses the official [UNICODE CLDR](#) JSON data and also it provides the [loadCldr](#) method to load the culture specific CLDR JSON data.

To go with the different culture other than [English](#), follow the below steps.

- Install the **CLDR-Data** package by using the below command (it installs the CLDR JSON data). To know more about CLDR-Data refer the [CLDR-Data](#) link.

`

```
npm install cldr-data --save
```

`

Once the package installed, you can find the culture specific JSON data under the location **/node_modules/cldr-data**.

- Now import the installed CLDR JSON data into the **app.component.ts** file.
- Now use the [loadCldr](#) method to load the culture specific CLDR JSON data from the installed location to **app.component.ts** file.
- Calendar displayed **Sunday** as the first day of week based on default culture ("en-US"). If you want to display the Calendar with loaded culture's first day of week, you need to import **weekdata.json** file from the **cldr-data/supplemental** as given in the code example.

```
`typescript
```

```
//import the loadCldr from ej2-base
```

```
import { loadCldr } from '@syncfusion/ej2-base';
```

```
declare var require: any;
```

```
loadCldr(
```

```
require('cldr-data/supplemental/numberingSystems.json'),
```

```
require('cldr-data/main/de/ca-gregorian.json'),
```

```
require('cldr-data/main/de/numbers.json'),
```

```
require('cldr-data/main/de/timeZoneNames.json'),
```

```
require('cldr-data/supplemental/weekdata.json')); // To load the culture based first day of week
```

`

The **Localization** library allows you to localize default text content of the Calendar. The Calendar component has static text for **today** feature that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

Locale keywords | Text

today | Name of the button to choose Today date.

- Before changing to a culture other than **English**, ensure that locale text for the concerned culture is loaded through **load** method of **L10n** class.

```
`typescript
```

```
//Load the L10n, loadCldr from ej2-base
```

```
import { loadCldr, L10n } from "@syncfusion/ej2-base";
```

//load the locale object to set the localized placeholder value

```
L10n.load({
  de: {
    calendar: {
      today:"heute"
    }
  }
});
`
```

- Set the culture by using the [locale](#) property. The below code example, initialize the Calendar component in **German** culture.

```
`typescript
import { Component } from '@angular/core';
//import the loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
declare var require: any;
loadCldr(
  require('cldr-data/supplemental/numberingSystems.json'),
  require('cldr-data/main/de/ca-gregorian.json'),
  require('cldr-data/main/de/numbers.json'),
  require('cldr-data/main/de/timeZoneNames.json')
);
@Component({
  selector: 'app-root',
  template: `
<!-- Sets the value, locale -->
<ejs-calendar [value]='dateValue' locale='de'></ejs-calendar>`
})
export class AppComponent {
  public dateValue: Object = new Date();
  ngOnInit(): void {
    //loads the localization text/
    L10n.load({
```

```

'de': {
  'calendar': {
    today: "heute"
  }
}
});
}
constructor() {
}
}
,

```

The following example demonstrates the Calendar in **German** culture.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
import { loadCldr, L10n } from '@syncfusion/ej2-base';
// Here we have referred local json files for preview purpose
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as timeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);
@Component({
  imports: [

    CalendarModule //declaration of ej2-angular-calendars module into
    NgModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <!-- Sets the value, locale -->
    <ejs-calendar [value]='dateValue' locale='de'></ejs-calendar>`
})
export class AppComponent {
  public dateValue: Object = new Date();
  ngOnInit(): void {
    /*loads the localization text*/
    L10n.load({
      'de': {
        'calendar': {
          today: "heute"
        }
      }
    });
  }
});

```

```

    }
    constructor() {
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Right-To-Left

The Calendar supports right-to-left functionality for languages like Arabic, Hebrew to displays the text in the right-to-left direction. Use [enableRtl](#) property to set the RTL direction.

The following example demonstrates the Calendar in Arabic culture with `enableRtl` property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
import { loadCldr, L10n } from '@syncfusion/ej2-base';
// Here we have referred local json files for preview purpose
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as timeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);
@Component({
  imports: [

    CalendarModule //declaration of ej2-angular-calendars module into
    NgModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <!-- Sets the value, locale,enableRtl -->
    <ejs-calendar [value]='dateValue' locale='ar'
enableRtl='true'></ejs-calendar>
  `
})
export class AppComponent {
  public dateValue: Object = new Date();
  ngOnInit(): void {
    L10n.load({
      'ar': {
        'calendar': { today: "اليوم" }
      }
    });
  }
  constructor() {
  }
}

```



```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customization in Angular Calendar component

Calendar allows you to customize the entire appearance by using the custom CSS and [renderDayCell](#) event to customize the each day cell.

This following section demonstrates how to disable, highlights the specific dates in the Calendar.

Disable Weekends

You can disable the weekends of every month in a Calendar by using the [renderDayCell](#) event. The `isDisabled` argument from this event allows you to define whether the date to be disabled or not.

Set [isDisabled](#) to true to disable the date value.

The following example demonstrates how to disable weekends of every month.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [

    CalendarModule //declaration of ej2-angular-calendars module into
    NgModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <!-- Bind renderDayCell event to customize and disable the day
    cell. -->
    <ejs-calendar [value]='dateValue'
    (renderDayCell)='disabledDate($event)'></ejs-calendar>
  `
})
export class AppComponent {
  public dateValue: Date = new Date();
  constructor() {
  }
  disabledDate(args : any): void {
    if (args.date.getDay() === 0 || args.date.getDay() === 6) {
      //set 'true' to disable the weekends
      args.isDisabled = true;
    }
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Day Cell Format

You can highlight the specific dates by adding the custom CSS or element to the day cell by using [renderDayCell](#).

Below is the list of classes that provides the flexible way to customize the Calendar component.

Class Name	Description
---	---
e-calendar	Applied to Calendar.
e-header	Applied to header.
e-title	Applied to title.
e-icon-container	Applied to previous and next icon container.
e-prev	Applied to previous icon.
e-next	Applied to next icon.
e-weekend	Applied to weekend dates.
e-other-month	Applied to other month dates.
e-day	Applied to each day cell.
e-selected	Applied to selected dates.
e-disabled	Applied to disabled dates.

The following example highlights the world health date (7th April every year) and world forest day (21st March every year) in a Calendar by using the custom icon and tooltip.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    CalendarModule //declaration of ej2-angular-calendars module into
    NgModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./style.css'],
  template: `
    <!-- To customize day calendar appearance -->
```

```

<!-- Refer the "special,highlight-day" class details in
"styles.css"-->
<ejs-calendar [value]='dateValue'
(renderDayCell)='customDates($event)'></ejs-calendar>
))
export class AppComponent {
    public dateValue: Date = new Date('3/7/2017');
    constructor() {
    }
    customDates(args: any): void {
        let span: HTMLElement;
        //defines the custom HTML element to be added.
        span = document.createElement('span');
        //Use "e-icons" class name to load Essential JS 2 built-in icons.
        span.setAttribute('class', 'e-icons highlight-day');
        if (+args.date.getDate() === 7 && +args.date.getMonth() === 3) {
            //append the span element to day cell.
            args.element.appendChild(span);
            //set the custom tooltip to the special dates.
            args.element.setAttribute('title', 'World health day!');
            //Use "special" class name to highlight the special dates, which you
            can refer in "styles.css".
            args.element.className = 'special';
        }
        if (+args.date.getDate() === 21 && +args.date.getMonth() === 2) {
            args.element.appendChild(span);
            args.element.className = 'special';
            //set the custom tooltip to the special dates.
            args.element.setAttribute('title', 'World forest day');
        }
    }
}

```

STYLE.CSS

```

/* Example - styles */
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    top: 45%;
    left: 45%;
}
#element {
    display: block;
    height: 350px;
}
#wrapper {
    width: 250px;
    margin: 0 auto;
}

```

```

/* Custom - styles */
.e-icons.highlight-day {
  /* csslint allow: adjoining-classes*/
  color: red;
}
.e-icons.highlight-day {
  /* csslint allow: adjoining-classes*/
  margin-top: -13px;
  display: block;
  margin-left: 4px;
}
.e-icons.highlight-day,
/* csslint allow: adjoining-classes*/
.e-icons.highlight-day:before {
  /* csslint allow: adjoining-classes*/
  color: rgb(0, 0, 255);
}
.e-icons.highlight-day:before {
  /* csslint allow: adjoining-classes*/
  content: "\e190";
  vertical-align: middle;
  margin-right: 3px;
  font-size: 4px;
  position: relative;
  top: 1px;
  font-weight: normal;
}
.special.e-day.e-ripple-style {
  /* csslint allow: adjoining-classes*/
  font-weight: bold;
  color: red;
}
.e-selected .e-icons.highlight-day:before {
  /* csslint allow: adjoining-classes*/
  color: #fff;
}
@font-face {
  font-family: 'e-icons';
  src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKIAAAwAgTlMvMjciQ6oAAAEoAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcX
fOQAAAKAAAAg4aGVhZBLt+DYAADQAAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAYAAAA
wbG9jYQukCgIAAAIkAAAAAGmlheHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbw
AAAZoAAAArAABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAADAAABAAAAQAAlbrm7l8
PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAAD8wPzAAAACAACAAAAAAAAAAAAEAAAAAQIAAwAAAA
AAgAAAAoACgAAP8AAAAAAAAAAQPqAZAABQAAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAWgPzAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAD6AAAA+gAAAPoAAD6AAAA+gAAAPoAAD6AAAAAAAAAgAAAAAMAAAAUAMAAQA
AABQABABeAAADgAIAAIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhEuEa4ZD//wAAAAAAAAAAAAA
AAABAA4ADgAOABYAFgAYAAAAQACAAYABAADAAGABwAKAAkABQALAAAAAAAAAB4AQABaQYB5gJ
kAnoCjgKwA8oEHAAAAAIAAAAAA+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D
+lwFpAcP+s24BTf6qbgAAAAEAAAAAAAA+oD6gALAAATCQEXCQEHCEHCQEnCQF4AYgBiGP+eAGIY/54/nh
jAYj+eApr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAAD6gOkAAMABwALAAA3IRUhESEVIREhFSE
VA9b8KgPW/CoDlvwq6I0B64wB640AAAAEAAAAAAAA+oD4QCaAAABMx8aHQEPDjEPAh8bIT8bNS8SPxs
CAA0aGhgMDAsLCwoKCgkJCQgHBwYGBgUEBAMCAGECawUFBggICQoLCwwMDg0GAgEBAGIDBAMIBiI
dHh0cHBoZFhUSEACFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQEAWIBAQMEDg0NDAs
LCQkJBwYGBAMCAQEBAgidBAQFBQYGBwgICakJCgoKCwsLDawMGRoD4gMEBwQFBQYGBwgICakKCgs
LDawNDQ4ODxAQEBEWFxYWFhYVFRQUEXIRERAOFxMLCggIBgYFBgQMDAwNDg4QDxERERIJCQkKCQk

```

```

JFRQJQCQJQJQJQJEhERERAPDw4NDQsMBwqFBgYICQkKDAwODw8RERMTEXUUFhUWfXyYWFxEQEBAPDg4
NDQwMCwsKCgkICAgHBgYFBQQEBQAAAAAAwAAAAAD8wPzAEEAZQDFAAABMx8FFREzHwYdAg8GIS8
GPQI/BjMlKwEvBT0CPwUzNzMfBR0CDWUrAi8FPQI/BTMnDw8ffz8XLxcPBgI+BQQDAwMCAT8EBAM
DAwIBAQIDAwMEBP7cBAQDAwMCAQECAwMDBAQ/PwQEAwMDAgEBAgMDAwQE0AUEAwMDAgEBAgMDAwQ
FfAUEAwMDAgEBAgMDAwQFvRsbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRocHR4eHyAgISI
iISAgHx4eHRSbGRcWFRMREA4LCQgFAwEBAwUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY
BAgIDBAQE/rMBAQEDAwQEBGgEBAQDAgIBAQBEBAgIDBAQEaAQEBAMDAQEB0AECaWMDBAVoBAQDAwM
CAeUBAgIEAwQEaAUEAwMDAgEBAgMDAwQFaAQEAwQCAGELERMVFhcZGhwdHh4fICaHiIhICAFHh4
dGxsZFxyVEExEQDgsJCAUDAQEDBQcKCw4QERMVFhcZGxsDhH4fICaHiIhICAFHh4dHBoZFxyVEExE
QDgsKBwUDAQEDBQcKCw4AAAAIAAAAAA9MD6QALAE8AAAE0AQcuASc+ATceAQEHBgcNjgYPAQYWHwE
FGBchDgEfAR4BPWEWHwEeAtSBmjY/ATY3Fxy2PwE2Ji8BNjQnNz4BLwEuAQ8BJi8BLgErASIGaps
BY0tKYwICY0pLY/7WEy4nfAkRBWQEAWdQANqBwMEZAURCXWfnLhMBDgnICg4BEy4mfQkRBGQFAwh
pAwNpCAMFZAQSCCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBLIYMhhsBHIrgg
FAzIfE4QJDAwJhBQeMgQGCK4IEgZSGDIYUGYSCK4IBQMjHxOECQwMAAEAAAAAAwED6gAFAAAJAic
JAQEbAef+FhoBzf4za+v+FF4VHwHMAc0AAAAAAQAAAAADAQPqAAUAAAEXCQEHAQLLHf4zaC0a/hY
D6x7+M/40HwHrAAEAAAAAA/MD8wALAAATCQEXCQE3CQEnCQENAY7+cmQBjwGPZP5yAY5k/nH+cQO
P/nH+cWQBjv5yZAGPAY9k/nEBjwAAAwAAAAAD8wPzAEEAgQEBAALdW4rAS8dPQE/DgUVDw4BPw4
7AR8dBRUfHTsBPx09AS8dKwEPHQL1DQ0ODg4PDw8QEBAQERERERUUFBQTEXITEREREBAPDw0ODAw
LCwkJCACGBgQEAgIBAgIEAwUFBgYHBwkICQoCYgECAGQDBQUGBgcHCQgJcV3QDQ0ODg4PDw8QEBA
QERERERUUFBQTEXITEREREBAPDw0ODAwLCwkJCACGBgQEAgL8fgIDBQUHCAkKCwwNDg8PERESEXQ
UFRYWFhgXGBKZGRoaGRkZGBcYFhYWFRQUEXIREQ8PDg0MCwoJCACFBQMCAgMFBQcICQoLDA0ODw8
RERITFBQVfHhYWGbcYGRkZGhoZGRkYFxfGWFhYVFBQTEhERDw8ODQwLCgkIBwUFAwLFCgkICQcHBgY
FBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETEhMTFBQUFREREREQEBAQDw8PDg40DQ31ERE
RERAQEABAPDw8ODg4NDQIwCgkICQcHBgYFBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETEhM
TFBQUFRoZGRkYFxfGWFhYVFBQTEhERDw8ODQwLCgkIBwUFAwICAwUFBwgJCgsMDQ4PDxEREhMUFBU
WFhYYFxfGZGRkaGhkZGRGxGBYWFhUUFBMSEREPDw4NDAsKCQgHBQUdAgIDBQUHCAkKCwwNDg8PERE
SEXQUFRYWFhgXGBkZGQAAQAAAAAD6gPqAEMAABMhHw8RDw8hLw8RPw6aAswNDgWMDAsKCggIBwU
FAWIBAQCIDBQUHCAgKCsMDAwODf00DQ4MDAwLDCgoICACFBQMCAQECAwUFBwgICgoLDAwMDgPrAQI
DBQUHCAgKCsLDA0NDv00Dg0NDAsLCgoICACFBQMCAQECAwUFBwgICgoLCwwNDQ4CzA4NDQwLCwo
KCAGHBQUdAgAAABIA3gABAAAAAEEEEAAAAABAAAAAABAA0AAQABAAAAAAACAAcAdgABAAAAAA
DAA0AFQABAAAAAAEEAA0AIgABAAAAAAFAAsALwABAAAAAAGAA0AOGABAAAAAAKACwArwABAAA
AAAAALABIAcWADAAEECQAAAAIAhQADAAEECQABABoAhwADAAEECQACAA4AoQADAAEECQADABoArwA
DAAEECQAEABoAyQADAAEECQAFABYA4wADAAEECQAGABoA+QADAAEECQAKAFgBEwADAAEECQALACQ
BayBlLWL1jb25zLW1ldHJvUmVndWxhcmtUtaWNvbnMtbnVW0cm9lLW1ljb25zLW1ldHJvVmVyc2l2bviA
xLjBlLW1ljb25zLW1ldHJvRm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z1c2l2bviBNZXRYbyBTdHV
kaW93d3cuc3luY2Z1c2l2bvi5jb20AIABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAFIAZQBnAHU
AbABhAHIAZQAtAGkAYwBvAG4AcwAtAG0AZQB0AHIAbwBlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgB
vAFYAZQBvAHMAaQBvAG4AIAIAAxAC4AMABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAEYAbwBuAHQ
AIAABnAGUAbgBlAHIAIAYQB0AGUAZAAgAHUAAcwBpAG4AZwAgAFMAeQBvAGMAZgBlAHMAaQBvAG4AIAIAB
NAGUAdABYAG8AIAIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBvAGMAZgBlAHMAaQBvAG4ALgBjAG8
AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAa
BCwEMAQ0AB2hvbWUtMDElQ2xvc2UtaWNvbnMhbnVudS0wMQR1c2VyB0JUX2luZm8PU2V0dGluZl9
BbmRyb2lkDWN0ZXZyb24tcmlnaHQMY2hldnJvbi1sZWZ0CE1UX0NsZWfYDE1UX0p1bmttYWlscwR
zdG9wAAA=) format('truetype');
font-weight: normal;
font-style: normal;
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Highlight Weekends

You can highlight the weekends of every month in a Calendar by using the [renderDayCell](#) event. The following example demonstrates how to highlight the weekends of every month.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [

    CalendarModule //declaration of ej2-angular-calendars module into
    NgModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <!-- Bind renderDayCell event to customize and highlight the
    weekend of every month. --->
    <ejs-calendar [value]='dateValue'
    (renderDayCell)='highlightWeekend($event)'></ejs-calendar>
  `
})
export class AppComponent {
  public dateValue: Date = new Date();
  constructor() {
  }
  highlightWeekend(args : any): void {
    if (args.date.getDay() === 0 || args.date.getDay() === 6) {
      // To highlight the week end of every month
      args.element.classList.add('e-highlightweekend');
    }
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Add the external button in the Calendar popup](#)
- [How to skip a month in Calendar](#)
- [How to change the first day of week](#)
- [How to customize the Calendar day header](#)

Calendar views in Angular Calendar component

The Calendar has the following pre-defined views that provides a flexible way to navigate back and forth to select the date.

Use the [start](#) property to change the default view of the Calendar.

View	Description
---	---
month (default)	Displays the days in a month
year	Displays the months in a year
decade	Displays the years in a decade

The following example demonstrates how to set the **year** as the start view of the Calendar.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    CalendarModule //declaration of ej2-angular-calendars module into
    NgModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <!-- Sets the value, start-->
    <ejs-calendar [value]='dateValue' start='Year'></ejs-calendar>
  `
})
export class AppComponent {
  public dateValue: Object = new Date();
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

View Restriction

Calendar view navigation can be restricted by defining the [start](#) and [depth](#) property that allows you to select the date from that view.

By defining the start and depth property with the different view, drill-down and drill-up views navigation can be limited to the user. Calendar views will be drill-down up to the view which is set in **depth** property and drill-up up to the view which is set in **start** property.

Always the depth view has to be smaller than the start view.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [

    CalendarModule //declaration of ej2-angular-calendars module into
    NgModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <!-- Sets the value, start and depth-->
    <ejs-calendar #ejCalendar [value]='dateValue' start='Decade'
    depth='Year'></ejs-calendar>
  `
})
export class AppComponent {
  public dateValue: Object = new Date();
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

You can restrict the calendar's drill down navigation by defining the [start](#) and [depth](#) property with same view that allows to select the date on that view itself.

The following example demonstrates how to select the dates in **year** view.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [

    CalendarModule //declaration of ej2-angular-calendars module into
    NgModule
  ],
```



```

standalone: true,
  selector: 'app-root',
  template: `
    <!-- Sets the value, start and depth"-->
    <ejs-calendar #ejCalendar [value]='dateValue' start='Year'
depth='Year'></ejs-calendar>
  `
})
export class AppComponent {
  public dateValue: Object = new Date();
  constructor() {
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Accessibility in Angular Calendar component

The Calendar component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Calendar component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

```
| Axe-core Accessibility Validation |  |
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Web accessibility defines a way to make web content and web applications more accessible to disabled people. It especially helps the dynamic content change and advanced user interface controls developed with Ajax, HTML, JavaScript, and related technologies.

Calendar provides built-in compliance with the [WAI-ARIA](#) specifications. WAI-ARIA supports is achieved through the attributes like `aria-label`, `aria-selected`, `aria-disabled`, `aria-activedescendant` applied for navigation buttons, disable and active day cells.

It helps to provide the information about the widget for assistive technology to the disabled person in the screen reader. Calendar component contains grid as role and grid cell for each day cell

- **Aria-label** : attribute provides the text label for an object for the previous and next month element. It helps the screen reader object to read for the assistive purpose.
- **Aria-selected** : attribute indicates the currently selected date of the Calendar component.
- **Aria-disabled** : attribute indicates the disabled state of this Calendar component.
- **Aria-activedescendent** : attribute helps in managing the current active child of the Calendar component.
- **Role** : attributes gives assistive technologies information about how to handle each element in a widget.
- **Grid-cell** : attributes define the individual cell that can be focusable and selectable.

Keyboard Interaction

You can use the following keys to interact with the Calendar.

The component implements the keyboard navigation support by following the [WAI-ARIA practices](#)

It supports the below list of shortcut keys.

```
| Press | To do this |
| --- | --- |
```

- | Upper Arrow | Focus the previous week date. |
- | Down Arrow | Focus the next week date. |
- | Left Arrow | Focus the previous date. |
- | Right Arrow | Focus the next date. |
- | Home | Focus the first date in the month. |
- | End | Focus the last date in the month. |
- | Page Up | Focus the same date in the previous month. |
- | Page Down | Focus the same date in the next month. |
- | Enter | Select the currently focused date. |
- | Shift + Page Up | Focus the same date in the previous year. |
- | Shift + Page Down | Focus the same date in the next year. |
- | Control + Upper Arrow | Moves into the inner level of view like month-year, year-decade |
- | Control + Down Arrow | Moves out from the depth level view like decade-year, year-month |
- | Control + Home | Focus the starting date in the current year. |
- | Control + End | Focus the ending date in the current year. |

To focus the Calendar component use the `alt+t` keys.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component, HostListener, ViewChild } from '@angular/core';
@Component({
  imports: [

    CalendarModule //declaration of ej2-angular-calendars module into
NgModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-calendar #ejCalendar [value]='dateValue'></ejs-calendar>
  `
})
export class AppComponent {
  @ViewChild('ejCalendar') ejCalendar: any;
  public dateValue: Date = new Date();
  @HostListener('document:keyup', ['$event'])
  handleKeyboardEvent(event: KeyboardEvent) {
    if (event.altKey && event.keyCode === 84) {
      // press alt+t to focus the control.
      this.ejCalendar.element.querySelectorAll('.e-content
table')[0].focus();
    }
  }
}
```

```

    }
    constructor() {
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Ensuring accessibility

The Calendar component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Calendar component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Calendar component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Islamic calendar in Angular Calendar component

In addition to the Gregorian calendar, the Calendar control supports displaying the Islamic calendar (Hijri calendar). **Islamic calendar** or **Hijri calendar** is a **lunar calendar** consisting of 12 months in a year of 354 or 355 days. To know more about Islamic calendar, please refer this [wikipedia](#).

Also, it consists of all Gregorian calendar functionalities as like min and max date, week number, start day of the week, multi selection, enable RTL, start and depth view, localization, highlight and customize the specific dates.

By default, calendar mode is in **Gregorian**. You can enable the Islamic mode by setting the **calendarMode** as **Islamic**. Also, need to import and injecting the **IslamicService** module into the **providers** section of root **NgModule** or component class from **ej2-angular-calendars** as shown below.

```
import { IslamicService, Calendar } from '@syncfusion/ej2-angular-calendars';
```

The following example demonstrates how to display the Islamic Calendar (Hijri Calendar).

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { CalendarModule, IslamicService } from '@syncfusion/ej2-angular-calendars';
import { Component } from '@angular/core';
import { IslamicService } from '@syncfusion/ej2-angular-calendars';
import { addClass } from '@syncfusion/ej2-base';
@Component({
  imports: [

    CalendarModule //declaration of ej2-angular-calendars module into
    NgModule
  ]
})

```

```

    ],
    providers: [IslamicService],
    standalone: true,
    selector: 'app-root',
    providers: [IslamicService],
    template: `
        <!-- Sets the isMultiSelection and values properties-->
        <ejs-calendar (renderDayCell)="onLoad($event)"
        (change)="onValueChange($event)" calendarMode='Islamic' class='e-
        customStyle'></ejs-calendar>`
    })
    export class AppComponent {
        constructor() {
        }
        onValueChange(args: any) {}
        onLoad(args: any) {
            /*Date need to be disabled*/
            if (args.date.getDate() === 12 || args.date.getDate() === 17 ||
args.date.getDate() === 22) {
                args.isDisabled = true;
            }
            /*Dates need to be customized*/
            if (args.date.getDate() === 13) {
                let span: HTMLElement;
                span = document.createElement('span');
                args.element.children[0].className += 'e-day sf-icon-cup
highlight';
                addClass([args.element], ['special', 'e-day', 'dinner']);
                args.element.setAttribute('data-val', 'Dinner !');
                args.element.appendChild(span);
            }
            if (args.date.getDate() === 23) {
                let span: HTMLElement;
                span = document.createElement('span');
                args.element.children[0].className += 'e-day sf-icon-start
highlight';
                span.setAttribute('class', 'sx !');
                //use the imported method to add the multiple classes to the
given element
                addClass([args.element], ['special', 'e-day', 'holiday']);
                args.element.setAttribute('data-val', 'Holiday !');
                args.element.appendChild(span);
            }
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Style appearance in Angular Calendar component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the background color for the Calendar

Use the following CSS to customize the background color and border for the Calendar.

```
`css
/ To specify background color and border /
.e-calendar {
background-color: peachpuff;
border: 3px solid red;
}
`
```

Customizing the Calendar date elements on hovering

Use the following CSS to customize the date elements on hovering in the Calendar.

```
`css
/ To specify background color, color, and border /
.e-calendar .e-content td:hover span.e-day, .e-calendar .e-content td:focus span.e-day, .e-bigger.e-small
.e-calendar .e-content td:hover span.e-day, .e-bigger.e-small .e-calendar .e-content td:focus span.e-day
{
background-color: red;
border: 2px solid;
color: #212529;
}
`
```

Customizing the border of date cell grid

Use the following CSS to add the border to the date cell grid.

```
`css
/ To specify border /
.e-calendar .e-content span.e-day, .e-bigger.e-small .e-calendar .e-content span.e-day {
border: 1px solid;
}
`
```

Customizing the Calendar title

Use the following CSS to customize the Calendar title.

```
`css
```

/ To specify color and font size /

```
.e-calendar .e-header .e-title, .e-bigger.e-small .e-calendar .e-header .e-title {  
color: black;  
font-size: 20px;  
}  
,
```

Customizing the previous and next icon

Use the following CSS to customize the previous and next icon.

```
`css
```

/ To specify color and border /

```
.e-calendar .e-header span, .e-bigger.e-small .e-calendar .e-header span {  
border: 1px solid;  
color: chocolate;  
}  
,
```

Customizing the footer button

Use the following CSS to customize the footer button.

```
`css
```

/ To specify background color, color, and border-color /

```
.e-calendar .e-btn.e-today.e-flat.e-primary, .e-calendar .e-css.e-btn.e-today.e-flat.e-primary {  
background-color: red;  
border-color: black;  
color: black;  
}  
,
```

Customizing the selected date cell grid

Use the following CSS to customize the selected date cell grid in Calendar.

```
`css
```

/ To specify background color and color /

```
.e-calendar .e-content td.e-focused-date.e-today span.e-day {  
background-color: maroon;  
color: #fff;  
}  
,
```

Customizing the content header in Calendar

Use the following CSS to customize the content header in Calendar.

```
`css

/ To specify background /

.e-calendar .e-content thead, .e-bigger.e-small .e-calendar .e-content thead {
background: aquamarine;
}
`
```

How To

Json data binding with calendar in Angular Calendar component

In most of the real cases, the model data will be available with JSON format only. Here we have showcased Calendar component by setting JSON string to value property. In this JSON, we have used ISO formatted date string which is frequently used date format to get proper date and time value without any misreading.

Also our Calendar component supports the ISO formatted date value, so parsed JSON value can be directly set to Calendar model.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
export interface User {
    selectedDate: Date;
}
export interface JSONUser {
    selectedDate: string;
}
@Component({
    imports: [

        CalendarModule //declaration of ej2-angular-calendars module into NgModule
    ],
    standalone: true,
    selector: 'app-root',
    template: `
        <!-- To Render Calendar -->
        <ejs-calendar id="calendar" [(value)]='user.selectedDate'
        (change)='onChange($event)'></ejs-calendar>
        <div class="valuestring">
            <b>User model</b>: <br/>{{user | json }}
            <br/><br/>
            <b>JSON Data</b>: <br/>{{ model_result }}
            <br/><br/>
        </div>`
    })
export class AppComponent {
    public user?: User | any;
```



```

    public JSONData: JSONUser = JSON.parse('{ "selectedDate": "2018-12-18T08:56:00+00:00"}');
    public model_result: string = JSON.stringify(this.JSONData);
    constructor() {}
    public ngOnInit() {
        this.user = this.JSONData;
    }
    onChange(args: any) {
        this.JSONData.selectedDate = args.value;
        this.model_result = JSON.stringify(this.JSONData);
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Two way binding in Angular Calendar component

The following example demonstrates how to achieve **two-way binding** by binding the **value** to the first Calendar component by using property binding and binding the model data using **ngModel** by using model binding to the Calendar component. The **value** of the Calendar will get change, when there is any change in the property value or model value.

The two-way binding can also be achieved only by using **property binding** or **model binding** in the Calendar component.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
import { CalendarComponent } from '@syncfusion/ej2-angular-calendars';
@Component({
  imports: [

    CalendarModule,
    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <!-- two-way binding using the value binding and model binding in
the Calendar -->
    <ejs-calendar id="firstcalendar" #ejCalendar
[(value)]='value'></ejs-calendar>
    <ejs-calendar id="secondcalendar" #ejCalendars
[(ngModel)]='value'></ejs-calendar>
  `
})
export class AppComponent {

```

```

    value: Date;
    constructor() {
        this.value = new Date('1/1/2020');
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Custom event emitter in Angular Calendar component

The **two-way binding** in Calendar can also be achieved using the custom event binding and property binding in the controls present in two different components. To create custom event, we need to create an instance of **event emitter**.

In the following example, **property binding** is used to share the data from the parent component to the child component using **@input** directive and **custom event binding** is used to share the data from the child component by using **@output** directive.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
import { CalendarComponent } from '@syncfusion/ej2-angular-calendars';
@Component({
  imports: [ CalendarModule ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div class="parentelement">
    <span><h4>Parent Component</h4></span>
    <div class="datevalue">
    <ejs-calendar id="calendar" #calendar (change)="deposit()"
    [value]="value" width="200px"></ejs-calendar>
    </div>
    </div>
    <child [xvalue]="value" (valueChange)="valuecheck($event)"> </child>
    `
})
export class ParentComponent {
  @ViewChild('calendar')
  public calendar?: CalendarComponent;
  value: Date;
  constructor() {
    this.value = new Date("2/1/2020");
  }
  deposit() {
    this.value = ((this.calendar) as CalendarComponent).value;
  }
  valuecheck(args: any) {

```

```

        this.value = args;
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Set clear button in calendar in Angular Calendar component

The following steps illustrate how to configure **clear** button in Calendar UI.

1. On **created** event of Calendar add the required elements to have clear button. Here we have used div with Essential JS 2 Button component.
2. Use the **e-footer** class to the div tag to act as the footer.
3. Use button to clear the selected date from the calendar.
4. Bind the required event handler to the button tag to clear the value.

Below is the code example.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
import { CalendarComponent } from '@syncfusion/ej2-angular-calendars';
@Component({
  imports: [
    CalendarModule,
    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./style.css'],
  template: `
    <!-- Bind created event to add the clear button --->
    <ejs-calendar #ejCalendar (created)='onCreate()'></ejs-calendar>
  `
})
export class AppComponent {
  @ViewChild('ejCalendar') ejCalendar?: CalendarComponent;
  onCreate() {
    let clearBtn: HTMLElement = document.createElement('button');
    let footerElement: HTMLElement = document.getElementsByClassName('e-footer-container')[0] as HTMLElement;
    //creates the custom element for clear button
    clearBtn.className = 'e-btn e-clear e-flat';
    clearBtn.textContent = 'Clear';
    footerElement.prepend(clearBtn);
  }
}

```

```

        (this.ejCalendar as CalendarComponent
    ).element.appendChild(footerElement);
    let proxy = this;
    // custom click handler to update the value property with null
    values.
    (document.querySelector('.e-footer-container .e-clear') as Element
    ).addEventListener('click', function() {
        (proxy.ejCalendar as CalendarComponent ).value = null as any;
    })
    };
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Show dates of other months in Angular Calendar component

The following example demonstrates how to show dates in other months.

The below styles changes the Calendar's other month dates to visible state from its hidden state.

```

`css
.e-calendar .e-content tr.e-month-hide {
display: table-row;
}
.e-calendar .e-content td.e-other-month>span.e-day {
display: inline-block;
}
.e-calendar .e-content td.e-month-hide,
.e-calendar .e-content td.e-other-month {
pointer-events: auto;
touch-action: auto;
}
`

```

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
imports: [

```

```

    CalendarModule //declaration of ej2-angular-calendars module into
    NgModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <!-- To show other months date refer the "styles.css". -->
    <ejs-calendar [value]='dateValue'></ejs-calendar>
  `
})
export class AppComponent {
  public dateValue: Date = new Date();
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Select a sequence of dates in calendar in Angular Calendar component

The following example demonstrates how to select the week dates of chosen date in the Calendar using [values](#) property, when [isMultiSelection](#) property is enabled. Methods of Moment.js is used in this sample for calculating the start and end of week from the selected date.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
import { CalendarComponent } from '@syncfusion/ej2-angular-calendars';
import * as moment from 'moment';
@Component({
  imports: [

    CalendarModule //declaration of ej2-ng-calendars module into
    NgModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./style.css'],
  template: `
    <!-- Rendering the Calendar with Multi selection option-->
    <div class="wrapelement">
      <ejs-calendar #ejCalendar id="calendar" isMultiSelection='true'
      (change)="onChange($event)"></ejs-calendar>
    </div>
    <div class="btncontainer e-btn-group e-vertical">
      <input type="radio" id="workweek" name="week" value="workweek"
      (click)="workWeek()" />
      <label class="e-btn" for="workweek">Work Week</label>
      <input type="radio" id="week" name="week" value="week"
      (click)="week()" />
    </div>
  `
})

```

```

        <label class="e-btn" for="week">Week</label>
      </div>
    ))
  export class AppComponent {
    @ViewChild('ejCalendar') CalendarInstance?: CalendarComponent;
    /*selected current week dates when click the button*/
    workWeek() {
      if (this.CalendarInstance?.element.classList.contains('week')) {
        this.CalendarInstance?.element.classList.remove('week')
      }
      this.CalendarInstance?.element.classList.add('workweek');
    }
    week() {
      if (this.CalendarInstance?.element.classList.contains('workweek')) {
        this.CalendarInstance?.element.classList.remove('workweek')
      }
      this.CalendarInstance?.element.classList.add('week');
    }
    onChange(args: any) {
      var startOfWeek = moment(args.value).startOf('week');
      var endOfWeek: any = moment(args.value).endOf('week');
      if (this.CalendarInstance?.element.classList.contains('workweek')) {
        this.getWeekArray(startOfWeek.day(1), endOfWeek.day(5), this);
      } else if
    (this.CalendarInstance?.element.classList.contains("week")) {
      this.getWeekArray(startOfWeek, endOfWeek, this);
    }
  }
  getWeekArray(startOfWeek: any, endOfWeek: number, obj: this) {
    var days = [];
    var day = startOfWeek;
    while (day <= endOfWeek) {
      days.push(day.toDate());
      day = day.clone().add(1, 'd');
    }
    (obj.CalendarInstance as CalendarComponent).values = days;
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Skip a month in calendar in Angular Calendar component

The following example demonstrates how to skip a month in a Calendar while clicking the previous and next icon. Here we have used the [navigated](#) event to skip a month using [NavigateTo](#) method.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'

```

```

import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
import { CalendarComponent } from '@syncfusion/ej2-angular-calendars';
@Component({
  imports: [

    CalendarModule //declaration of ej2-angular-calendars module into NgModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-calendar #ejCalendar (navigated)='onNavigate($event)'></ejs-
calendar>
  `
})
export class AppComponent {
  @ViewChild('ejCalendar') ejCalendar?: CalendarComponent;
  //skips a month while clicking previous and next icon in month view.
  onNavigate(args : any):void {
    let date: Number | any;
    if ((<HTMLInputElement>(event as
Event).currentTarget).classList.contains('e-next')) {
      //incrementing the month while clicking the next icon
      date = new Date(args.date.setMonth(args.date.getMonth() + 1));
    }
    if ((<HTMLInputElement>(event as
Event).currentTarget).classList.contains('e-prev')) {
      //decrementing the month while clicking the previous icon
      date = new Date(args.date.setMonth(args.date.getMonth() - 1));
    }
    if (args.view == 'month') {
      this.ejCalendar?.navigateTo('month' as any, new Date('' + date));
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customize the calendar day header in Angular Calendar component

You can change the format of the day that to be displayed in header using [dayHeaderFormat](#) property. By default, the format is **Short**.

You can find the possible formats on below.

Name	Description
Short	Sets the short format of day name (like Su) in day header.

| **Narrow** | Sets the single character of day name (like S) in day header. |

| **Abbreviated** | Sets the min format of day name (like Sun) in day header. |

| **Wide** | Sets the long format of day name (like Sunday) in day header. |

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, ViewChild } from '@angular/core';
import { CalendarComponent } from '@syncfusion/ej2-angular-calendars';
import { DropDownListComponent, ChangeEventArgs } from '@syncfusion/ej2-angular-dropdowns';
@Component({
  imports: [

    DropDownListModule,
    CalendarModule //declaration of ej2-angular-calendars module into
    NgModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./style.css'],
  template: `
    <div id="container">
      <div id="calendar">
        <ejs-calendar #default dayHeaderFormat='Short'></ejs-
calendar>
      </div>
      <div id="format">
        <label class="custom-input-label">Header Format
Types</label>
        <ejs-dropdownlist id="dayformat" #select
[dataSource]='formatData' [(value)]=value [fields]='fields'
[placeholder]='waterMark' (change)='formatHandler($event)'></ejs-
dropdownlist>
      </div>
    </div>
  `
})
export class AppComponent {
  @ViewChild('default')
  public calendarObj?: CalendarComponent;
  @ViewChild('select')
  public dayHeaderFormat?: DropDownListComponent;
  // define the JSON of data
  public formatData: Object[] = [
    { Id: 'Short', Label: 'Short' },
    { Id: 'Narrow', Label: 'Narrow' },
    { Id: 'Abbreviated', Label: 'Abbreviated' },
    { Id: 'Wide', Label: 'Wide' },
  ];
  public fields: Object = { text: 'Label', value: 'Id' };
  public waterMark: string = 'Select format type';
```



```

    public value: string = 'Short';
    public formatHandler(args: ChangeEventArgs): void {
        (this.calendarObj as CalendarComponent).dayHeaderFormat =
args.value as any;
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Render the calendar with week numbers in Angular Calendar component

You can enable the `weekNumber` in Calendar by using the [weekNumber](#)

property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [

      CalendarModule //declaration of ej2-angular-calendars module into
NgModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
      <!-- Sets the weekNumber -->
      <ejs-calendar [value]='dateValue' weekNumber='true'></ejs-calendar>
  `
})
export class AppComponent {
    public dateValue: Date = new Date();
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Change the first day of week in Angular Calendar component

The Calendar provides an option to change the first day of the week by using the [firstDayOfWeek](#) property.

Day of the week starts from 0(Sunday) to 6(Saturday).

By default, the first day of week will be based on culture.

The following example demonstrates the Calendar with **Tuesday** as first day of the week.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [

    CalendarModule //declaration of ej2-angular-calendars module into
    NgModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <!-- Sets the firstDayOfWeek -->
    <ejs-calendar [value]='dateValue'
[firstDayOfWeek]='startWeek'></ejs-calendar>
  `
})
export class AppComponent {
  public dateValue: Date = new Date();
  public startWeek: number = 2;
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Card

Getting started with Angular Card component

This section explains how to create a simple **Card** using Styles, and how to configure the structure for the header section, Horizontal, action buttons, content section, and configure its available functionalities in Angular using Angular quickstart.

Dependencies

The Card Component is pure CSS component so no specific dependencies to render the card.

`js

|-- @syncfusion/ej2-layouts

,

Setup Angular Environment

You can use [Angular CLI](#) to setup your Angular applications.

To install Angular CLI use the following command.

```
`bash
npm install -g @angular/cli
`
```

Create an Angular Application

Start a new Angular application using below Angular CLI command.

```
`bash
ng new my-app
cd my-app
`
```

Installing Syncfusion Card package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(`>=20.2.36`) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-layouts](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-layouts --save
`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-layouts@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-layouts@ngcc --save
`
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-layouts:"20.2.38-ngcc"
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding CSS reference

The following CSS files are available in `../node_modules/@syncfusion` package folder.

This can be referenced in `[src/styles.css]` using following code.

```
`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-layouts/styles/material.css';
`
```

Adding a simple Card

- Add the HTML `div` element with `e-card` class into your `index.html`.

[src/index.html]

```
`html
<div class = "e-card">
Sample Card
</div>
`
```

Adding a header to the card

You can create cards with a header in a specific structure. For adding header you need to create `div` element and add `e-card-header` class.

- You can include heading inside the card header by adding an `div` element with `e-card-header-caption` class, and also content will be added by adding element with `e-card-content`. For detailed information, refer to the [Header and Content](#).

```
`html
<div class = "e-card">           --> Root Element
<div class="e-card-header">      --> Root Header Element
<div class="e-card-header-caption"> --> Root Heading Element
<div class="e-card-header-title"></div> --> Heading Title Element
</div>
<div class="e-card-content"></div> --> Card content Element
</div>
</div>
`
```

- Now, run the application in the browser using the following command.

```
npm start
```

Output will be as follows:

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [

  ],
  standalone: true,
  selector: 'app-container',
  template: `
    <div tabindex="0" class="e-card" id="basic">
      <div class="e-card-header">
        <div class="e-card-header-caption">
          <div class="e-card-title">Advanced UWP</div>
        </div>
      </div>
      <div class="e-card-content">
        Communicating with Windows 10 and Other Apps, the second in
a five-part series written by Succinctly series
        author Matteo Pagani. To download the complete white paper,
and other papers in the series, visit
        the White Paper section of Syncfusion's Technology Resource
Portal.
      </div>
    </div>
  `
})
export class AppComponent {
  @ViewChild('element') element: any;
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [How to add a header and content](#)

Header content in Angular Card component

Header

The Card can be created with header title, sub title and images. For adding header you need to create `div` element with the class `e-card-header` added.

Card provides below elements and corresponding class definitions to include header.

Elements | Description

`e-card-header-caption` | To group the title and subtitle within the header which acts as wrapper.

`e-card-header-title` | Main title text with in the header.

`e-card-sub-title` | A sub-title within the header.

`e-card-header-image` | To include heading image within the header.

`e-card-corner` | To add rounded corner for the image.

Title and Subtitle

For adding header to the Card , you need to create wrapper `div` element with `e-card-header-caption` class.

- Place the `div` element with `e-card-header-title` class inside the header caption for adding main title.
- Place the `div` element with `e-card-sub-title` class inside the header caption element for adding sub-title.

Image

Card header has an option for adding images in the header. It is aligned with either before or after the header based on the HTML element positioned in the header structure.

- The header image can be added by creating a `div` element with `e-card-header-image` class which can be placed before or after the header caption wrapper element.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [

  ],
  standalone: true,
  selector: 'app-container',
  template: `
    <div style="margin: 50px;">
      <div tabindex="0" class="e-card">
        <div class="e-card-header">
          <div class="e-card-header-image football e-card-
corner"></div>
          <div class="e-card-header-caption">
            <div class="e-card-header-title"> Laura Callahan</div>
```

```

        <div class="e-card-sub-title">Sales Coordinator and
Representative</div>
        </div>
    </div>
</div>
<div style="margin-left: 50px;margin-top:30px">
<div tabindex="0" class="e-card">
    <div class="e-card-header">
        <div class="e-card-header-caption">
            <div class="e-card-header-title"> Laura Callahan</div>
            <div class="e-card-sub-title">Sales Coordinator and
Representative</div>
        </div>
        <div class="e-card-header-image football"></div>
    </div>
</div>
</div>
`
}))
export class AppComponent {
    @ViewChild('element') element: any;
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Content

Content in Card holds texts, images, links and all possible HTML elements. Its adaptable within the Card root element.

- Create a `div` element with the class `e-card-content`.
- Place content `div` element in the Card root element or within any Card inner elements.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    ],
  standalone: true,
  selector: 'app-container',
  template: `
    <div tabindex="0" class="e-card">
      <div class="e-card-header">
        <div class="e-card-header-image football"></div>
        <div class="e-card-header-caption">

```

```

        <div class="e-card-header-title"> Laura Callahan</div>
        <div class="e-card-sub-title">Sales Coordinator and
Representative</div>
      </div>
    </div>
    <div class="e-card-content">
      Laura received a BA in psychology from the University of
Washington. She has also completed a course in business French. She reads
and writes French.
    </div>
  </div>
})
export class AppComponent {
  @ViewChild('element') element: any;
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Card image in Angular Card component

Images

The Card supports to include images within the elements, you can add image as direct element anywhere inside card root by adding the `e-card-image` class to `div` element. Using the class defined, you can write CSS styles to load images to that element.

By default, card images occupies full width of its parent element.

```
`html
```

```

<div class = "e-card">
  <div class="e-card-image">
  </div>
</div>
`

```

Title

Card image is supported to include a title or caption for the image. By default, Title is placed over the image on left-bottom position with overlay.

```
`html
```

```

<div class = "e-card">
  <div class="e-card-image">
    <div class="e-card-title"></div>
  </div>
`

```



```
</div>
```

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [

  ],
  standalone: true,
  selector: 'app-container',
  template: `
    <div class="e-card">
      <div class="e-card-image">
        <div class="e-card-title">JavaScript </div>
      </div>
      <div class="e-card-content"> JavaScript Succinctly was written
to give readers an accurate, concise examination of JavaScript objects and
their supporting nuances, such as complex values, primitive values, scope,
inheritance, the head object, and more. </div>
    </div>
  `
})
export class AppComponent {
  @ViewChild('element') element: any;
  ngAfterViewInit() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Divider

Divider used to separate the elements inside the card. You can add divider inside the card elements to separate it.

- Place the `div` element with `e-card-separator` class inside the card element for adding a divider.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [

  ],
```

```

standalone: true,
selector: 'app-container',
template: `
  <div tabindex="0" class="e-card" id="basic">
    <div class="e-card-title">Explore Cities</div>
    <div class="e-card-separator"></div>
    <div class="e-card-content">
      Sydney is a city on the east coast of Australia. Sydney is
the capital city of New South Wales. About four million people
      live in Sydney which makes it the biggest city in Oceania.
    </div>
    <div class="e-card-separator"></div>
    <div class="e-card-content">
      New York City has been described as the cultural, financial,
and media capital of the world, and exerts a significant impact
      upon commerce and etc.,
    </div>
    <div class="e-card-separator"></div>
    <div class="e-card-content">
      Malaysia is one of the Southeast Asian countries, on a
peninsula of the Asian continent, to a certain extent; it can be recognized
      as part of the Asian continent.
    </div>
  </div>
`
  ))
export class AppComponent {
  @ViewChild('element') element: any;
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [How to customize the card image title position](#)

Action buttons in Angular Card component

You can include Action buttons within the Card and customize them. Action button is a `div` element with `e-card-actions` class followed by button tag or anchor tag within the card root element.

- For adding action buttons you can create button or anchor tag with `e-card-btn` class within the card action element.

`html

```
<div class = "e-card">
```

```
<div class="e-card-actions">
```

```
<button class="e-card-btn"></button>
```

```
<a href="#"></a>
```

```
</div>
```

```
</div>
```

```
,
```

Vertical

By default, action buttons positioned in horizontal alignment , and also it can be aligned to show in vertical alignment by adding `e-card-vertical` class.

```
`html
```

```
<div class = "e-card">
```

```
<div class="e-card-actions e-card-vertical">
```

```
<button class="e-card-btn">More</button>
```

```
<a href="#">Share</a>
```

```
</div>
```

```
</div>
```

```
,
```

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [

    ],
  standalone: true,
  selector: 'app-container',
  template: `
    <div style="margin: 50px;">
      <div class="e-card" style="max-width:400px">
        <div class="e-card-header-title">Eiffel Tower</div>
        <div class="e-card-content">
          The Eiffel Tower is acknowledged as the universal symbol of
          Paris and France.
        </div>
        <div class="e-card-actions">
          <button class="e-card-btn">
            
          </button>
          <button class="e-card-btn">
            
          </button>
          <button class="e-card-btn">
```

```

        
        </button>
    </div>
</div>
</div>
<div style="margin-left: 50px;">
    <div class="e-card" style="max-width:400px">
        <div class="e-card-header-title">Eiffel Tower</div>
        <div class="e-card-content">
            The Eiffel Tower is acknowledged as the universal symbol
of Paris and France.
        </div>
        <div class="e-card-actions e-card-vertical">
            <button class="e-card-btn">LIKE</button>
            <button class="e-card-btn">SHARE</button>
        </div>
    </div>
</div>
    `
    })
    export class AppComponent {
        @ViewChild('element') element: any;
    }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [How to integrate other component inside the card](#)

Horizontal in Angular Card component

By default, all the card elements are aligned vertically one after the other as in the DOM.

You can achieve the element to align horizontally as well by adding the class `e-card-horizontal` in the root card element.

Stacked cards

- An horizontally aligned card can push a specific column to align vertical using `e-card-stacked` class.

This will align the stacked section vertically aligned differentiating from horizontal layout.

Class | Description

`e-card-horizontal` | To align card elements horizontally.

`e-card-stacked` | To align elements vertically within the horizontal layout.

```
`html
<div tabindex="0" class="e-card e-card-horizontal">
 --> Aligned in horizontal
<div class="e-card-stacked">    --> Aligned in horizontal
// Inside the element all are aligned vertical directions
</div>
</div>
`
```

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [

    ],
  standalone: true,
  selector: 'app-container',
  template: `
    <div style="margin: 50px;display: flex;flex-direction: row;justify-
content: center;">
      <div tabindex="0" class="e-card e-card-horizontal"
style="width:400px">
        
        <div class="e-card-stacked">
          <div class="e-card-header">
            <div class="e-card-header-caption">
              <div class="e-card-header-title">Philips
Trimmer</div>
            </div>
          </div>
          <div class="e-card-content">
            Powered by the innovative DuraPower Technology which
optimizes power consumption, Philips trimmers are designed to last longer
than 4 ordinary trimmers.
          </div>
        </div>
      </div>
    </div>
  `
})
export class AppComponent {
  @ViewChild('element') element: any;
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Style in Angular Card component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on user preference.

Customizing the card

Use the following CSS to customize the card properties.

```
`css
.e-card {
background-color: aqua;
padding-left: 20px;
margin-bottom: 20px;
}
`
```

Customizing the Header element

Use the following CSS to customize the Header element properties.

```
`css
.e-card .e-card-header {
font-family: cursive;
font-style: italic;
}
`
```

Customizing the card content

Use the following CSS to customize the card content properties.

```
`css
.e-card .e-card-content {
font-size: 20px;
color: gray;
line-height: initial;
font-weight: normal;
}
`
```

Divider used to separate the elements inside the card

Use the following CSS to customize the Divider used to separate the elements inside the card properties.

```
`css
```

```
.e-card .e-card-separator {  
padding-bottom: 30px;  
}  
`
```

Including image within card element

Use the following CSS to Include image within card element.

```
`css  
.e-card .e-card-image {  
background-image: url(images.png);  
background-color: yellow;  
height: 160px;  
}  
`
```

Including a title or caption for the image

Use the following CSS to Include a title or caption for the image.

```
`css  
.e-card .e-card-image .e-card-title {  
font-family: cursive;  
font-style: italic;  
}  
`
```

To include heading image within the header

Use the following CSS to Include heading image within the header.

```
`css  
.e-card .e-card-header .e-card-header-image {  
height: 48px;  
width: 48px;  
}  
`
```

Customizing the Header main title

Use the following CSS to Customize the Header main title.

```
`css  
.e-card .e-card-header .e-card-header-caption .e-card-header-title {  
font-size: large;  
`
```

```
color: aquamarine;  
}
```

,

Customizing the Header subtitle

Use the following CSS to Customize the Header subtitle.

```
`css  
.e-card .e-card-header .e-card-header-caption .e-card-sub-title {  
font-size: 20px;  
font-variant: all-petite-caps;  
}
```

,

Including action buttons or anchor tags

Use the following CSS to Include action buttons or anchor tags.

```
`css  
.e-card .e-card-actions .e-card-btn {  
padding-left: 20px;  
background-color: wheat;  
}
```

,

To align card elements horizontally

Use the following CSS to align card elements horizontally.

```
`css  
.e-card .e-card-horizontal {  
margin: auto;  
width: inherit;  
}
```

,

To align elements vertically within the horizontal layout

Use the following CSS to align elements vertically within the horizontal layout.

```
`css  
.e-card .e-card-horizontal .e-card-stacked {  
justify-content: flex-start;  
margin: initial;  
}
```


How To

Customize the card image title position in Angular Card component

Card Image titles are placed as always Bottom-Left Corner only, You can manually customize to placing titles anywhere over the image by adding styles.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, ViewChild } from '@angular/core';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
@Component({
  imports: [

    ],
  standalone: true,
  selector: 'app-container',
  template: `
    <div>
      <div class="e-card">
        <div class="e-card-image">
          <div class="e-card-title">Node.js</div>
        </div>
        <div class="e-card-content">
          Node.js is a wildly popular platform for writing web
          applications that has revolutionized web development in many ways, enjoying
          support across the open source community as well as
          industry.
        </div>
      </div>
    </div>
    <div style="Margin: 5px 0;width:300px">
      <select id="title_position">
        <option value="bottom-left">BottomLeft</option>
        <option value="top-left">TopLeft</option>
        <option value="top-right">TopRight</option>
        <option value="bottom-right">BottomRight</option>
      </select>
    </div>
  `
})
export class AppComponent {
  @ViewChild('element') element: any;
  ngAfterViewInit () {
    // initialize DropDownList component
    let dropDownListObject: DropDownList = new DropDownList({
      placeholder: "Select Position",
      change: this.changed,
    });
    // render initialized DropDownList
    dropDownListObject.appendTo('#title_position');
  }
  public changed(e: any) {
    let cardEle = document.querySelector('.e-card');
```

```

    let titleEle = cardEle?.querySelector('.e-card-image .e-card-
title');
    titleEle!.className = ' '
    titleEle?.classList.add('e-card-title');
    titleEle?.classList.add('e-card-'+e.value);
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Integrate other component inside the card in Angular Card component

You can integrate any component inside the card element. Here ListView component is placed inside the card for showcasing the To-Do list.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, ViewChild } from '@angular/core';
import { ListView } from '@syncfusion/ej2-lists';
@Component({
  imports: [

  ],
  standalone: true,
  selector: 'app-container',
  template: `
    <div style="margin: 50px;">
      <!--element which is going to render the Card-->
      <div tabindex="0" class="e-card" id="basic">
        <div class="e-card-title">To-Do List</div>
        <div class="e-card-separator"></div>
        <div class="e-card-content">
          <div id='element'></div>
        </div>
      </div>
    </div>
  `
})
export class AppComponent {
  @ViewChild('element') element: any;
  ngAfterViewInit () {
    //define the array of JSON
    let todoList: { [key: string]: Object }[] = [
      { todoList: 'Pay Bills' },
      { todoList: 'Call Chris' },
      { todoList: 'Meet Andrew' },
      { todoList: 'Visit Manager' },
      { todoList: 'Customer Meeting' },
    ];
    //Initialize ListView component
  }
}

```

```
let listViewInstance: ListView = new ListView({
  dataSource: todoList,
  //map the appropriate columns to fields property
  fields: { text: 'todoList' },
  showCheckBox: true,
});
//Render initialized ListView
listviewInstance.appendTo("#element");
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Carousel

Getting started with Angular Carousel component

This section explains how to create a simple [Angular Carousel](#), and demonstrate the basic usage of the Carousel module in an Angular environment.

Dependencies

The list of dependencies required to use the Carousel module in your application is given below:

```
`javascript
|-- @syncfusion/ej2-angular-navigations
|-- @syncfusion/ej2-angular-base
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-buttons
,`
```

Setup Angular environment

You can use [Angular CLI](#) to setup your Angular applications. To install Angular CLI use the following command.

```
,`
npm install -g @angular/cli
,`
```

Create an Angular application

Start a new Angular application using below Angular CLI command.

```
,`
ng new my-app
```

```
cd my-app
```

```
,
```

Installing Syncfusion Carousel Package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(`>=20.2.36`) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-navigations](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-navigations --save
```

```
,
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-navigations@ngcc](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-navigations@ngcc --save
```

```
,
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
```

```
@syncfusion/ej2-angular-navigations:"20.2.38-ngcc"
```

```
,
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding Carousel module

Import Carousel module into Angular application(`app.module.ts`) from the package

`@syncfusion/ej2-angular-navigations`.

```
`javascript
```

```
import { NgModule } from "@angular/core";
```

```
import { BrowserModule } from "@angular/platform-browser";
// Imported Syncfusion Carousel module from navigations package.
import { CarouselModule } from "@syncfusion/ej2-angular-navigations";
import { AppComponent } from "./app.component";
@NgModule({
  imports: [BrowserModule, CarouselModule], // Registering EJ2 Carousel Module.
  declarations: [AppComponent],
  bootstrap: [AppComponent],
})
export class AppModule {}
`
```

Adding Syncfusion Carousel component

Modify the template in `app.component.ts` file with `ejs-carousel` to render the Carousel component.

```
`javascript
import { Component } from "@angular/core";
@Component({
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
<div class="control-container">
<ejs-carousel>
<e-carousel-items>
<e-carousel-item>
<ng-template #template>
<figure class="img-container">

<figcaption class="img-caption">Cardinal</figcaption>
</figure>
</ng-template>
</e-carousel-item>
<e-carousel-item>
<ng-template #template>
<figure class="img-container">
```

```

<figcaption class="img-caption">Kingfisher</figcaption>
</figure>
</ng-template>
</e-carousel-item>
<e-carousel-item>
<ng-template #template>
<figure class="img-container">

<figcaption class="img-caption">Keel-billed-toucan</figcaption>
</figure>
</ng-template>
</e-carousel-item>
<e-carousel-item>
<ng-template #template>
<figure class="img-container">

<figcaption class="img-caption">Yellow-warbler</figcaption>
</figure>
</ng-template>
</e-carousel-item>
<e-carousel-item>
<ng-template #template>
<figure class="img-container">

<figcaption class="img-caption">Bee-eater</figcaption>
</figure>
</ng-template>
</e-carousel-item>
</e-carousel-items>
```

```

</ejs-carousel>
</div>`,
})
export class AppComponent {}
`

```

Adding CSS reference

Add Carousel component's styles as given below in `style.css`.

```

`css
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
`

```

Running the application

Run the application in the browser using the following command:

```

`
ng serve
`

```

The following example shows a basic `Carousel` component.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [ CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel>
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">

```

```

        
        <figcaption class="img-caption">Kingfisher</figcaption>
    </figure>
</ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Yellow-warbler</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Bee-eater</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
</e-carousel-items>
</ejs-carousel>
</div>`,
    })
export class AppComponent {}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note: You can also explore our [Angular Carousel example](#) that shows you how to configure the Carousel in Angular.

Populating items in Angular Carousel component

In the Carousel, slides can be rendered in two ways as follows,

- Populating items using carousel item
- Populating items using data source

Populating items using carousel item

When rendering the Carousel component using items binding, you can assign templates for each item separately or assign a common template to each item. You can also customize the slide transition interval for each item separately. The following example code depicts the functionality as item property binding.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [ ButtonModule, CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel>
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Kingfisher</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
        </e-carousel-items>
      </ejs-carousel>
    </div>`
})
@NgModule({
  imports: [BrowserModule, CarouselModule, ButtonModule],
  declarations: [AppComponent],
  bootstrap: [AppComponent],
})
export default AppModule;
```

```

        </figure>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <figure class="img-container">
          
          <figcaption class="img-caption">Yellow-warbler</figcaption>
        </figure>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <figure class="img-container">
          
          <figcaption class="img-caption">Bee-eater</figcaption>
        </figure>
      </ng-template>
    </e-carousel-item>
  </e-carousel-items>
</ejs-carousel>
</div>`,
  ))
export class AppComponent {}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Populating items using data source

When rendering the Carousel component using data binding, you can assign a common template only for all items using the [itemTemplate](#) property. You cannot set the interval for each item. The following example code depicts the functionality as data binding.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { CarouselModule } from '@syncfusion/ej2-angular-navigations';
import { Component } from '@angular/core';
@Component({
  imports: [ButtonModule, CarouselModule],
  standalone: true,
  selector: 'app-root',
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel [dataSource]="productItems">

```

```

        <ng-template #itemTemplate let-data>
            <div class="slide-content">{{ data.Title }}</div>
            <figure class="img-container">
                <img [src]="getImage(data.imageName)" alt="{{data.Name}}"
style="height:100%;width:100%;" />
                <figcaption class="img-caption">{{data.Name}}</figcaption>
            </figure>
        </ng-template>
    </ejs-carousel>
</div>`,
    })
    export class AppComponent {
        public productItems: Record<string, string | number>[] = [
            { ID: 1, Name: "Cardinal", imageName: 'cardinal' },
            { ID: 2, Name: "Kingfisher", imageName: 'hunei' },
            { ID: 3, Name: "Keel-billed-toucan", imageName: 'costa-rica' },
            { ID: 4, Name: "Yellow-warbler", imageName: 'kaohsiung' },
            { ID: 5, Name: "Bee-eater", imageName: 'bee-eater' }
        ];
        public getImage(bird: string): string {
            return
`https://ej2.syncfusion.com/products/images/carousel/${bird}.png`;
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Selection

The Carousel items will be populated from the first index of the Carousel items and can be customized using the following ways,

- Select an item using the property.
- Select an item using the method.

Select an item using the property

Using the [selectedIndex](#) property of the Carousel component, you can set the slide to be populated at the time of initial rendering else you can switch to the particular slide item.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
    imports: [ ButtonModule, CarouselModule],
    standalone: true,
    selector: "app-root",

```

```

template: `<!-- To Render Carousel. -->
<div class="control-container">
  <ejs-carousel [selectedIndex]="selectedIndex">
    <e-carousel-items>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Cardinal</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Kingfisher</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Yellow-warbler</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Bee-eater</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
    </e-carousel-items>
  </ejs-carousel>

```

```

    </div>`,
  })
  export class AppComponent {
    public selectedIndex: number = 3;
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Select an item using the method

Using the [prev](#) or [next](#) public method of the Carousel component, you can switch the current populating slide to a previous or next slide.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild } from '@angular/core';
import { CarouselComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [ ButtonModule, CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <button ej2-button cssClass="e-info"
    (click)="prevBtnClick()">Previous</button>
    <button ej2-button cssClass="e-info"
    (click)="nextBtnClick()">Next</button>
    <div class="control-container">
      <ejs-carousel #carousel>
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Kingfisher</figcaption>
              </figure>

```

```

        </ng-template>
      </e-carousel-item>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Yellow-warbler</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Bee-eater</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
    </e-carousel-items>
  </ejs-carousel>
</div>`,
  })
  export class AppComponent {
    @ViewChild("carousel") carousel!: CarouselComponent;
    public prevBtnClick() {
      this.carousel.prev();
    }
    public nextBtnClick() {
      this.carousel.next();
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Partial visible slides

The Carousel component supports to show one complete slide and a partial view of adjacent (previous and next) slides at the same time. You can enable or disable the partial slides using the [partialVisible](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [ ButtonModule, CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel>
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Kingfisher</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Yellow Warbler</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
        </e-carousel-items>
      </ejs-carousel>
    </div>`
})
@NgModule({
  imports: [BrowserModule, CarouselModule, ButtonModule],
  declarations: [AppComponent],
  bootstrap: [AppComponent],
})
export default AppModule;
```

```

        <figcaption class="img-caption">Yellow-warbler</figcaption>
      </figure>
    </ng-template>
  </e-carousel-item>
  <e-carousel-item>
    <ng-template #template>
      <figure class="img-container">
        
        <figcaption class="img-caption">Bee-eater</figcaption>
      </figure>
    </ng-template>
  </e-carousel-item>
</e-carousel-items>
</ejs-carousel>
</div>`,
  })
  export class AppComponent {}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Slide animation only applicable if the `partialVisible` is enabled.

The last slide will be displayed as a partial slide at the initial rendering when the `loop` and `partialVisible` properties are enabled.

The previous slide is not displayed at the initial rendering when the `loop` is disabled.

The following example code depicts the functionality of `partialVisible` and without `loop` functionalities.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [ ButtonModule, CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel>
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">

```



```

        
        <figcaption class="img-caption">Cardinal</figcaption>
    </figure>
</ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Kingfisher</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Yellow-warbler</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Bee-eater</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
</e-carousel-items>
</ejs-carousel>
</div>`,
    })
    export class AppComponent {}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Customizing partial slides size](#)

Navigators and indicators in Angular Carousel component

The navigators and indicators are used to transition the slides manually.

Navigators

Show or hide previous and next button

In navigators, the previous and next slide transition buttons are used to perform slide transitions manually. You can show/hide the navigators using the [buttonsVisibility](#) property. The possible property values are as follows:

- **Hidden** – the navigator's buttons are not visible.
- **Visible** – the navigator's buttons are visible.
- **VisibleOnHover** – the navigator's buttons are visible only when hovering over the carousel.

The following example depicts the code to show/hide the navigators in the carousel.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
import { CarouselButtonVisibility } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [ ButtonModule, CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel [buttonsVisibility]="buttonsVisibility">
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
```

```

        <figure class="img-container">
            
            <figcaption class="img-caption">Kingfisher</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Yellow-warbler</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Bee-eater</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
</e-carousel-items>
</ejs-carousel>
</div>`,
    ))
export class AppComponent {
    public buttonsVisibility: CarouselButtonVisibility = "Visible";
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Show previous and next button on hover

In the carousel, you can show the previous and next buttons only on mouse hover using the [buttonsVisibility](#) property. The following example depicts the code to show the navigators on mouse hover in the carousel.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
import { CarouselButtonVisibility } from '@syncfusion/ej2-angular-
navigations';
@Component({
  imports: [ ButtonModule, CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel [buttonsVisibility]="buttonsVisibility">
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Kingfisher</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
```

```

        
        <figcaption class="img-caption">Yellow-warbler</figcaption>
    </figure>
</ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Bee-eater</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
</e-carousel-items>
</ejs-carousel>
</div>`,
}))
export class AppComponent {
    public buttonsVisibility: CarouselButtonVisibility = "VisibleOnHover";
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Previous and next button template

Template options are provided to customize the previous button using [previousButtonTemplate](#) and the next button using [nextButtonTemplate](#). The following example depicts the code for applying the template to previous and next buttons in the carousel.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
import { CarouselButtonVisibility } from '@syncfusion/ej2-angular-navigations';
@Component({
    imports: [ ButtonModule, CarouselModule],
    standalone: true,
    selector: "app-root",
    template: `<!-- To Render Carousel. -->
    <div class="control-container">
        <ejs-carousel>
            <ng-template #previousButtonTemplate let-data>
                <button

```

```

        ejs-button
        cssClass="e-flat e-round"
        iconCss="e-icons e-chevron-left-double"
    ></button>
</ng-template>
<ng-template #nextButtonTemplate let-data>
    <button
        ejs-button
        cssClass="e-flat e-round"
        iconCss="e-icons e-chevron-right-double"
    ></button>
</ng-template>
<e-carousel-items>
    <e-carousel-item>
        <ng-template #template>
            <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
            </figure>
        </ng-template>
    </e-carousel-item>
    <e-carousel-item>
        <ng-template #template>
            <figure class="img-container">
                
                <figcaption class="img-caption">Kingfisher</figcaption>
            </figure>
        </ng-template>
    </e-carousel-item>
    <e-carousel-item>
        <ng-template #template>
            <figure class="img-container">
                
                <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
            </figure>
        </ng-template>
    </e-carousel-item>
    <e-carousel-item>
        <ng-template #template>
            <figure class="img-container">
                
                <figcaption class="img-caption">Yellow-warbler</figcaption>
            </figure>
        </ng-template>
    </e-carousel-item>
    <e-carousel-item>
        <ng-template #template>
            <figure class="img-container">

```

```

        
        <figcaption class="img-caption">Bee-eater</figcaption>
    </figure>
</ng-template>
</e-carousel-item>
</e-carousel-items>
</ejs-carousel>
</div>`,
    })
export class AppComponent {}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Indicators

Show or hide indicators

In indicators, the total slides and current slide state have been depicted. You can show/hide the indicators using the [showIndicators](#) property. The following example depicts the code to show/hide the indicators in the carousel.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [ ButtonModule, CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel [showIndicators]="showIndicators">
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">

```

```

        
        <figcaption class="img-caption">Kingfisher</figcaption>
    </figure>
</ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Yellow-warbler</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Bee-eater</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
</e-carousel-items>
</ejs-carousel>
</div>`,
    })
export class AppComponent {
    public showIndicators: Boolean = true;
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```


Indicators Template

Template option is provided to customize the indicators by using the [indicatorTemplate](#) property. The following example depicts the code for applying a template to indicators in the carousel.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [ CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel>
        <ng-template #indicatorsTemplate let-data>
          <div class="indicator" indicator-index="data.index"></div>
        </ng-template>
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Kingfisher</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
```

```

        
        <figcaption class="img-caption">Yellow-warbler</figcaption>
    </figure>
</ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Bee-eater</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
</e-carousel-items>
</ejs-carousel>
</div>`,
    })
    export class AppComponent {}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Showing preview of slide in indicator

You can customize the indicators by showing the preview image of each slide using the [indicatorTemplate](#) property. The following example depicts the code for showing the preview image using a template for indicators in the carousel.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [ CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <ejs-carousel>
      <ng-template #indicatorsTemplate let-data>
        <div class="indicator" indicator-index="data.index">
          <div class="preview-content">{{ getContent(data.index) }}</div>
        </div>
      </ng-template>
    <e-carousel-items>
      <e-carousel-item>
        <ng-template #template>

```

```

        <div class="slide-content">Slide 1</div>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <div class="slide-content">Slide 2</div>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <div class="slide-content">Slide 3</div>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <div class="slide-content">Slide 4</div>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <div class="slide-content">Slide 5</div>
      </ng-template>
    </e-carousel-item>
  </e-carousel-items>
</ejs-carousel>,
}))
export class AppComponent {
  public getContent(index: number): string {
    const slides: string[] = [
      "Slide 1",
      "Slide 2",
      "Slide 3",
      "Slide 4",
      "Slide 5",
    ];
    return slides[index];
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Indicators Types

Choose different types of indicators available using the [indicatorsType](#) property. The indicator types are categorized as follows:

- [Default Indicator](#)
- [Dynamic Indicator](#)
- [Fraction Indicator](#)
- [Progress Indicator](#)

Default Indicator

A default indicator in a carousel is a set of dots that indicate the current position of the slide in the carousel. The Default indicator can be achieved by setting the [indicatorsType](#) to `Default`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [ ButtonModule, CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel [autoPlay]="false" indicatorsType = "Default">
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Kingfisher</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Yellow-warbler</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
        </e-carousel-items>
      </ejs-carousel>
    </div>`
})
```

```

        </figure>
      </ng-template>
    </e-carousel-item>
  <e-carousel-item>
    <ng-template #template>
      <figure class="img-container">
        
        <figcaption class="img-caption">Bee-eater</figcaption>
      </figure>
    </ng-template>
  </e-carousel-item>
</e-carousel-items>
</ejs-carousel>
</div>`,
  })
  export class AppComponent {
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Dynamic Indicator

A dynamic indicator in a carousel provides visual cues or markers that dynamically change or update to indicate the current position. The Dynamic indicator can be achieved by setting the [indicatorsType](#) to **Dynamic**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [ ButtonModule, CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel [autoPlay]="false" indicatorsType = "Dynamic">
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
        </e-carousel-items>
      </ejs-carousel>
    </div>
  `
})

```

```

        </figure>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <figure class="img-container">
          
          <figcaption class="img-caption">Kingfisher</figcaption>
        </figure>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <figure class="img-container">
          
          <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
        </figure>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <figure class="img-container">
          
          <figcaption class="img-caption">Yellow-warbler</figcaption>
        </figure>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <figure class="img-container">
          
          <figcaption class="img-caption">Bee-eater</figcaption>
        </figure>
      </ng-template>
    </e-carousel-item>
  </e-carousel-items>
</ejs-carousel>
</div>`,
  })
  export class AppComponent {
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';

```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Fraction Indicator

The fraction indicator type displays the current slide index and total slide count as a fraction. The Fraction indicator can be achieved by setting the [indicatorsType](#) to [Fraction](#).

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [ ButtonModule, CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel [autoPlay]="false" indicatorsType = "Fraction">
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Kingfisher</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
```

```

        
        <figcaption class="img-caption">Yellow-warbler</figcaption>
    </figure>
</ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Bee-eater</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
</e-carousel-items>
</ejs-carousel>
</div>`,
}))
export class AppComponent {
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Progress Indicator

The Progress Indicator type displays the current slide as a progress bar. The Progress indicator can be achieved by setting the [indicatorsType](#) to **Progress**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [ ButtonModule, CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel [autoPlay]="false" indicatorsType = "Progress">
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">

```



```

        
        <figcaption class="img-caption">Cardinal</figcaption>
    </figure>
</ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Kingfisher</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Yellow-warbler</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
<e-carousel-item>
    <ng-template #template>
        <figure class="img-container">
            
            <figcaption class="img-caption">Bee-eater</figcaption>
        </figure>
    </ng-template>
</e-carousel-item>
</e-carousel-items>
</ejs-carousel>
</div>`,
    ))
    export class AppComponent {
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Play button*Show or hide the play button*

In the carousel, [autoPlay](#) actions have been controlled by using the [showPlayButton](#) property in the user interface. When you enable this property, the slide transitions are controlled using this play and pause button. This property depends on the [buttonsVisibility](#) property. The following example depicts the code to show the play button in the carousel.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [ ButtonModule, CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel [showPlayButton]="showPlayButton">
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Kingfisher</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Keel-billed toucan</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
        </e-carousel-items>
      </ejs-carousel>
    </div>`
})
@NgModule({
  imports: [BrowserModule, CarouselModule, ButtonModule],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export default AppModule;
```

```

        <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
      </figure>
    </ng-template>
  </e-carousel-item>
  <e-carousel-item>
    <ng-template #template>
      <figure class="img-container">
        
        <figcaption class="img-caption">Yellow-warbler</figcaption>
      </figure>
    </ng-template>
  </e-carousel-item>
  <e-carousel-item>
    <ng-template #template>
      <figure class="img-container">
        
        <figcaption class="img-caption">Bee-eater</figcaption>
      </figure>
    </ng-template>
  </e-carousel-item>
</e-carousel-items>
</ejs-carousel>
</div>`,
  })
  export class AppComponent {
    public showPlayButton: Boolean = true;
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Play button template

Template option is provided to customize the play button by using the [playButtonTemplate](#) property. The following example depicts the code for applying a template to play Button in the carousel.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { CarouselComponent } from "@syncfusion/ej2-angular-navigations";
import { ButtonComponent } from "@syncfusion/ej2-angular-buttons";
import { Component, ViewChild } from "@angular/core";
@Component({
  imports: [ ButtonModule, CarouselModule],

```

```

standalone: true,
selector: "app-root",
template: `<!-- To Render Carousel. -->
<div class="control-container">
  <ejs-carousel [showPlayButton]="showPlayButton" #carousel>
    <ng-template #playButtonTemplate let-data>
      <button
        ejs-button
        cssClass="e-info playBtn"
        [content]="content"
        (click)="btnClick()"
        #playButton
      ></button>
    </ng-template>
  <e-carousel-items>
    <e-carousel-item>
      <ng-template #template>
        <figure class="img-container">
          
          <figcaption class="img-caption">Cardinal</figcaption>
        </figure>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <figure class="img-container">
          
          <figcaption class="img-caption">Kingfisher</figcaption>
        </figure>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <figure class="img-container">
          
          <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
        </figure>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <figure class="img-container">
          
          <figcaption class="img-caption">Yellow-warbler</figcaption>
        </figure>
      </ng-template>
    </e-carousel-item>
  </e-carousel-items>
</div>
`

```

```

        <ng-template #template>
            <figure class="img-container">
                
                <figcaption class="img-caption">Bee-eater</figcaption>
            </figure>
        </ng-template>
    </e-carousel-item>
</e-carousel-items>
</ejs-carousel>
</div>`,
    })
    export class AppComponent {
        @ViewChild("carousel") carousel!: CarouselComponent;
        @ViewChild("playButton") playButton!: ButtonComponent;
        public showPlayButton: Boolean = true;
        public content: string = "Pause";
        public btnClick() {
            if (this.carousel.autoPlay) {
                this.playButton.content = "Play";
                this.carousel.autoPlay = false;
            } else {
                this.playButton.content = "Pause";
                this.carousel.autoPlay = true;
            }
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Animations and transitions in Angular Carousel component

Animations

Fade animation

In Carousel, two built-in animations are provided for slide transitions. You can disable animation using the [animationEffect](#) property. By default, Slide animation is applied for the transition between slides.

The following demo depicts the example for fade animation,

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
import { CarouselAnimationEffect } from '@syncfusion/ej2-angular-
navigations';
@Component({
    imports: [ ButtonModule, CarouselModule],

```

```

standalone: true,
selector: "app-root",
template: `<!-- To Render Carousel. -->
<div class="control-container">
  <ejs-carousel [animationEffect]="carouselAnimation">
    <e-carousel-items>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Cardinal</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Kingfisher</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Yellow-warbler</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Bee-eater</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
    </e-carousel-items>
  </ejs-carousel>
</div>`

```

```

        </e-carousel-items>
      </ejs-carousel>
    </div>`,
  })
  export class AppComponent {
    public carouselAnimation: CarouselAnimationEffect = 'Fade';
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Custom animation

In Carousel, you can use customized animation effects for slide transitions using the [Custom](#) option of the [animationEffect](#) property and apply custom animation css via [cssClass](#) property.

The following demo depicts the example for **parallax** custom animation,

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
import { CarouselAnimationEffect } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [ CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel [animationEffect]="carouselAnimation"
cssClass="parallax">
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Kingfisher</figcaption>
              </figure>

```

```

        </ng-template>
      </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <figure class="img-container">
          
          <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
        </figure>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <figure class="img-container">
          
          <figcaption class="img-caption">Yellow-warbler</figcaption>
        </figure>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <figure class="img-container">
          
          <figcaption class="img-caption">Bee-eater</figcaption>
        </figure>
      </ng-template>
    </e-carousel-item>
  </e-carousel-items>
</ejs-carousel>
</div>`,
  })
  export class AppComponent {
    public carouselAnimation: CarouselAnimationEffect = 'Custom';
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Intervals between slides

Using the items property, you can set different intervals for each item to transition between slides. The default interval is **5000 ms** (5 seconds). The following example depicts the code for setting the different intervals between each item.

APP.COMPONENT.TS


```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [ ButtonModule, CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel>
        <e-carousel-items>
          <e-carousel-item [interval]="firstInterval">
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item [interval]="secondInterval">
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Kingfisher</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item [interval]="thirdInterval">
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item [interval]="fourthInterval">
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Yellow-warbler</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item [interval]="fifthInterval">
            <ng-template #template>
              <figure class="img-container">

```

```

        
        <figcaption class="img-caption">Bee-eater</figcaption>
    </figure>
</ng-template>
</e-carousel-item>
</e-carousel-items>
</ejs-carousel>
</div>`,
    })
export class AppComponent {
    public firstInterval: number = 3000;
    public secondInterval: number = 1000;
    public thirdInterval: number = 2000;
    public fourthInterval: number = 5000;
    public fifthInterval: number = 6000;
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note: Interval property can accept value in terms of milliseconds.

Auto play slides

In the carousel, all slides transitions are performed continuously after the specified or default intervals. You can enable or disable the auto slide transition using the [autoPlay](#) property. The following example depicts the code to enable or disable the auto slide transitions.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
    imports: [ ButtonModule, CarouselModule],
    standalone: true,
    selector: "app-root",
    template: `<!-- To Render Carousel. -->
    <div class="control-container">
        <ejs-carousel [autoPlay]="autoPlay">
            <e-carousel-items>
                <e-carousel-item>
                    <ng-template #template>
                        <figure class="img-container">
                            
                            <figcaption class="img-caption">Cardinal</figcaption>

```

```

        </figure>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <figure class="img-container">
          
          <figcaption class="img-caption">Kingfisher</figcaption>
        </figure>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <figure class="img-container">
          
          <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
        </figure>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <figure class="img-container">
          
          <figcaption class="img-caption">Yellow-warbler</figcaption>
        </figure>
      </ng-template>
    </e-carousel-item>
    <e-carousel-item>
      <ng-template #template>
        <figure class="img-container">
          
          <figcaption class="img-caption">Bee-eater</figcaption>
        </figure>
      </ng-template>
    </e-carousel-item>
  </e-carousel-items>
</ejs-carousel>
</div>`,
  })
export class AppComponent {
  public autoPlay: Boolean = true;
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';

```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Pause on hover

By default, Slide transitions are paused when hovering the mouse pointer over the Carousel element. You can enable or disable this functionality using the [pauseOnHover](#) property.

The following example depicts the code to play the slides when hovering the mouse pointer over the Carousel element.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [ ButtonModule, CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel [pauseOnHover]="pauseOnHover">
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Kingfisher</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
        </e-carousel-items>
      </ejs-carousel>
    </div>`
})
```

```

        <e-carousel-item>
          <ng-template #template>
            <figure class="img-container">
              
              <figcaption class="img-caption">Yellow-warbler</figcaption>
            </figure>
          </ng-template>
        </e-carousel-item>
        <e-carousel-item>
          <ng-template #template>
            <figure class="img-container">
              
              <figcaption class="img-caption">Bee-eater</figcaption>
            </figure>
          </ng-template>
        </e-carousel-item>
      </e-carousel-items>
    </ejs-carousel>
  </div>`,
  })
  export class AppComponent {
    public pauseOnHover: Boolean = false;
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Looping slides

In the carousel, slides transitions are repeated continuously when you reach the last slide by default. You can enable or disable the infinite slide transition using the [loop](#) property. The following example depicts the code to enable or disable the infinite slide transitions.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [ ButtonModule, CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel [loop]="loop">
        <e-carousel-items>

```

```

        <e-carousel-item>
            <ng-template #template>
                <figure class="img-container">
                    
                    <figcaption class="img-caption">Cardinal</figcaption>
                </figure>
            </ng-template>
        </e-carousel-item>
        <e-carousel-item>
            <ng-template #template>
                <figure class="img-container">
                    
                    <figcaption class="img-caption">Kingfisher</figcaption>
                </figure>
            </ng-template>
        </e-carousel-item>
        <e-carousel-item>
            <ng-template #template>
                <figure class="img-container">
                    
                    <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
                </figure>
            </ng-template>
        </e-carousel-item>
        <e-carousel-item>
            <ng-template #template>
                <figure class="img-container">
                    
                    <figcaption class="img-caption">Yellow-warbler</figcaption>
                </figure>
            </ng-template>
        </e-carousel-item>
        <e-carousel-item>
            <ng-template #template>
                <figure class="img-container">
                    
                    <figcaption class="img-caption">Bee-eater</figcaption>
                </figure>
            </ng-template>
        </e-carousel-item>
    </e-carousel-items>
</ejs-carousel>
</div>`,
    })
    export class AppComponent {
        public loop: Boolean = true;

```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Slide changing events

Using the [slideChanging](#) or [slideChanged](#) events of the Carousel component, you can perform sample end customization while the carousel items are switched.

The following demo depicts the example for carousel events,

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
import {
  SlideChangingEventArgs,
  SlideChangedEventArgs,
} from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [ ButtonModule, CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel (slideChanging)="onSlideChanging($event)"
(slideChanged)="onSlideChanged($event)">
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Kingfisher</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
        </e-carousel-items>
      </div>
```

```

        <ng-template #template>
            <figure class="img-container">
                
                <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
            </figure>
        </ng-template>
    </e-carousel-item>
    <e-carousel-item>
        <ng-template #template>
            <figure class="img-container">
                
                <figcaption class="img-caption">Yellow-warbler</figcaption>
            </figure>
        </ng-template>
    </e-carousel-item>
    <e-carousel-item>
        <ng-template #template>
            <figure class="img-container">
                
                <figcaption class="img-caption">Bee-eater</figcaption>
            </figure>
        </ng-template>
    </e-carousel-item>
</e-carousel-items>
</ejs-carousel>
</div>`,
    ))
export class AppComponent {
    public onSlideChanging(args: SlideChangingEventArgs): void {
        console.log(args.currentSlide); // You can customize the slide before
        changing.
    }
    public onSlideChanged(args: SlideChangedEventArgs): void {
        console.log(args.currentSlide); // You can customize the slide after
        changed.
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```


Disable touch swiping

In the carousel, you can swipe the carousel slides using touch actions by default. The swipe action can be enabled or disabled using the [enableTouchSwipe](#) property. The following example depicts the code to disable the swipe action for the slide.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [ ButtonModule, CarouselModule],
  standalone: true,
  selector: "app-root",
  template: `<!-- To Render Carousel. -->
    <div class="control-container">
      <ejs-carousel [enableTouchSwipe]="enableTouchSwipe">
        <e-carousel-items>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Cardinal</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Kingfisher</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
          <e-carousel-item>
            <ng-template #template>
              <figure class="img-container">
                
                <figcaption class="img-caption">Yellow Warbler</figcaption>
              </figure>
            </ng-template>
          </e-carousel-item>
        </e-carousel-items>
      </ejs-carousel>
    </div>`
})
@NgModule({
  imports: [BrowserModule, CarouselModule, ButtonModule],
  declarations: [AppComponent],
  bootstrap: [AppComponent],
})
export default AppModule;
```

```

        <figcaption class="img-caption">Yellow-warbler</figcaption>
      </figure>
    </ng-template>
  </e-carousel-item>
  <e-carousel-item>
    <ng-template #template>
      <figure class="img-container">
        
        <figcaption class="img-caption">Bee-eater</figcaption>
      </figure>
    </ng-template>
  </e-carousel-item>
</e-carousel-items>
</ejs-carousel>
</div>`,
  })
  export class AppComponent {
    public enableTouchSwipe: Boolean = false;
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Swipe Modes

In the carousel, the [swipeMode](#) property allows specifying whether the slide transition should occur while performing swiping via touch or mouse. The slide swiping is enabled or disabled using the bitwise operator.

The following are the different swipe modes available in the carousel:

- CarouselSwipeMode.Touch - Allows the user to slide the slides using touch actions.
- CarouselSwipeMode.Mouse - Allows the user to slide the slides using mouse actions.
- CarouselSwipeMode.Touch & CarouselSwipeMode.Mouse - Allows the user to slide the slides using both touch and mouse actions.
- ~CarouselSwipeMode.Touch & ~CarouselSwipeMode.Mouse - Disables both touch and mouse actions.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CarouselModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
import { CarouselSwipeMode } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [ ButtonModule, CarouselModule],

```

```

standalone: true,
selector: "app-root",
template: `<!-- To Render Carousel. -->
<div class="control-container">
  <ejs-carousel [swipeMode]="carouselSwipe">
    <e-carousel-items>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Cardinal</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Kingfisher</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Keel-billed-
toucan</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Yellow-warbler</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
      <e-carousel-item>
        <ng-template #template>
          <figure class="img-container">
            
            <figcaption class="img-caption">Bee-eater</figcaption>
          </figure>
        </ng-template>
      </e-carousel-item>
    </e-carousel-items>
  </ejs-carousel>
</div>
`

```

```

        </e-carousel-items>
      </ejs-carousel>
    </div>`,
  })
  export class AppComponent {
    public carouselSwipe: CarouselSwipeMode = CarouselSwipeMode.Touch &
    CarouselSwipeMode.Mouse;
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

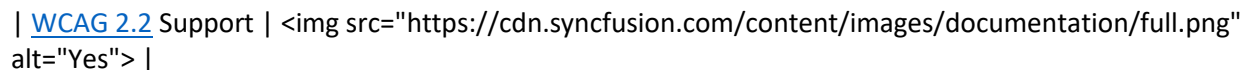
Accessibility in Angular Carousel component

The Carousel component has been designed, keeping in mind the [WAI-ARIA](#) specifications, and applying the WAI-ARIA roles, states and properties along with keyboard support for people who use assistive devices. WAI-ARIA accessibility support is achieved through attributes like `aria-roledescription`, `aria-label`, `aria-current`, `aria-live`, `aria-role` and `aria-hidden`. It provides information about elements in a document for assistive technology. The component implements keyboard navigation support by following the [WAI-ARIA practices](#) and has been tested in major screen readers.

The accessibility compliance for the Carousel component is outlined below.

| Accessibility Criteria | Compatibility |

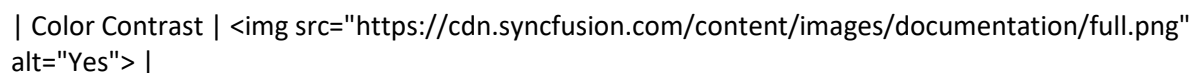
| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes" > |

| [Section 508](#) Support |  alt="Yes" > |

| Screen Reader Support |  alt="Yes" > |

| Right-To-Left Support |  alt="Yes" > |

| Color Contrast |  alt="Yes" > |

| Mobile Device Support |  alt="Yes" > |

| Keyboard Navigation Support |  alt="Yes" > |

| [Accessibility Checker](#) Validation |  alt="Yes" > |

```
| Axe-core Accessibility Validation |  |
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>
```

ARIA attributes

The Carousel component is designed by considering [WAI-ARIA](#) standard. Carousel is supported with ARIA Accessibility which is accessible by on-screen readers and other assistive technology devices. The following list of attributes is added to the Carousel.

| Roles and Attributes | Functionalities

----- -----	
aria-roledescription	The role description attribute has been added for the root element (Carousel) and each Carousel slide item (slide).
aria-label	Previous, next and play/pause buttons and all indicator elements.
aria-current	For the active item indicator element, <code>aria-current</code> is set to <code>true</code> .
aria-hidden	For all Carousel elements except the currently visible item, <code>aria-hidden</code> is set to <code>true</code> .
aria-live	For Carousel items element, when <code>autoPlay</code> is <code>true</code> , <code>aria-live</code> is set to <code>off</code> ; when <code>autoPlay</code> is <code>false</code> , <code>aria-live</code> is set to <code>polite</code> .
aria-role	For Carousel slide item, <code>aria-role</code> has been grouped.

Keyboard interaction

By default, keyboard navigation is enabled. This component implements keyboard navigation support by following the WAI-ARIA practices. Once focused on the active Carousel element, you can use the following key combination for interacting with the Carousel.

Key	Description
-----	-----
Alt + J	Keys to focus the Carousel component (done at application end).
Arrows	Keys to navigate between slides.
Home	To navigate to the first slide.
End	To navigate to the last slide.
Space	To play/pause the slide transitions.
Enter	To perform the respective action on its focus.
Tab	To Move focus through the interactive elements.
Shift + Tab	To Move focus through the interactive elements.

Ensuring accessibility

The Carousel component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Carousel component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Carousel component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Chart

Getting started with Angular Chart component

This section explains you the steps required to create a simple [Angular Chart](#)[Link to the Video](#) and demonstrate the basic usage of the Chart component in an Angular environment.

To get start quickly with Angular Chart using CLI and Schematics, you can check on this video:

Setup Angular Environment

You can use [Angular CLI](#) to setup your Angular applications.

To install Angular CLI use the following command.

```
`bash
npm install -g @angular/cli
`
```

Create an Angular Application

Start a new Angular application using below Angular CLI command.

```
`bash
ng new my-app
cd my-app
`
```

Installing Syncfusion Chart package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(`>=20.2.36`) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-charts](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-charts --save
```

```
,
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-charts@ngcc](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-charts@ngcc --save
```

```
,
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
```

```
@syncfusion/ej2-angular-charts:"20.2.38-ngcc"
```

```
,
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Registering Chart Module

Import Chart module into Angular application(`app.module.ts`) from the package `@syncfusion/ej2-angular-charts` [`src/app/app.module.ts`].

```
`typescript
```

```
import { NgModule } from '@angular/core';
```

```
import { BrowserModule } from '@angular/platform-browser';
```

```
// import the ChartModule for the Chart component
```

```
import { ChartModule } from '@syncfusion/ej2-angular-charts';
import { AppComponent } from './app.component';
@NgModule({
  //declaration of ChartModule into NgModule
  imports: [ BrowserModule, ChartModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

- Modify the template in `app.component.ts` file to render the `ej2-angular-charts` component

[src/app/app.component.ts].

```
`javascript
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  selector: 'app-container',
  // specifies the template string for the Charts component
  template: <ejs-chart id='chart-container'></ejs-chart>,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent { }
```

<!-- markdownlint-disable MD033 -->

Now use the `<code>app-container</code>` in the index.html instead of default one.

```
<app-container></app-container>
```

- Now run the application in the browser using the below command.

```
npm start
```


The below example shows a basic Charts.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  standalone: true,
  selector: 'app-container',
  // specifies the template string for the Chart component
  template: `<ejs-chart id="chart-container"></ejs-chart>`
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Module Injection

Chart component are segregated into individual feature-wise modules. In order to use a particular feature, you need to inject its feature service in the AppModule. In the current application, we are going to modify the above basic chart to visualize sales data for a particular year.

For this application we are going to use line series, tooltip, data label, category axis and legend feature of the chart. Please find relevant feature service name and description as follows.

- **LineSeriesService** - Inject this provider to use line series.
- **LegendService** - Inject this provider to use legend feature.
- **TooltipService** - Inject this provider to use tooltip feature.
- **DataLabelService** - Inject this provider to use datalabel feature.
- **CategoryService** - Inject this provider to use category feature.

These modules should be injected to the provider section as follows,

`javascript

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { ChartComponent } from '@syncfusion/ej2-angular-charts';
import { CategoryService, LegendService, TooltipService } from '@syncfusion/ej2-angular-charts';
```

```
import { DataLabelService, LineSeriesService } from '@syncfusion/ej2-angular-charts';
@NgModule({
  imports: [
    BrowserModule,
  ],
  declarations: [AppComponent, ChartComponent],
  bootstrap: [AppComponent],
  providers: [ CategoryService, LegendService, TooltipService, DataLabelService, LineSeriesService ]
})
`
```

Populate Chart with Data

This section explains how to plot below JSON data to the chart.

```
`javascript
export class AppComponent implements OnInit {
  public chartData: Object[];
  ngOnInit(): void {
    // Data for chart series
    this.chartData = [
      { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
      { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
      { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
      { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
      { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
      { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
    ];
  }
}
```

Add a series object to the chart by using [series](#) property. Now map the field names `month` and `sales` in the JSON data to the [xName](#) and [yName](#) properties of the series, then set the JSON data to [dataSource](#) property.

Since the JSON contains category data, set the [valueType](#) for horizontal axis to `Category`. By default, the axis `valueType` is `Numeric`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='month'
yName='sales' name='Sales'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  ngOnInit(): void {
    // Data for chart series
    this.chartData = [
      { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
      { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
      { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
      { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
      { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
      { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
    ];
    this.primaryXAxis = {
      valueType: 'Category'
    };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

The sales data are in thousands, so format the vertical axis label by adding

\$ as a prefix and **K** as a suffix to each label. This can be achieved by setting the

`\${value}K` to the [labelFormat](#) property of axis. Here, **{value}** act as a placeholder

for each axis label.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='month'
yName='sales' name='Sales'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public primaryYAxis?: Object;
  public chartData?: Object[];
  ngOnInit(): void {
    this.chartData = [
      { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
      { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
      { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
      { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
      { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
      { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
    ];
    this.primaryXAxis = {
      valueType: 'Category'
    };
    this.primaryYAxis = {
      labelFormat: '$ {value}K'
    };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Add Chart Title

You can add a title using [title](#) property to the chart to provide quick information to the user about the data plotted in the chart.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LegendService, TooltipService, DataLabelService,
LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LegendService, TooltipService,
DataLabelService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis'
[title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='month'
yName='sales' name='Sales'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public primaryYAxis?: Object;
  public title?: string;
  ngOnInit(): void {
    // Title for chart
    this.title = 'Sales Analysis';
    this.chartData = [
      { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
      { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
      { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
      { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
      { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
      { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
    ];
    this.primaryXAxis = {
      valueType: 'Category'
    };
    this.primaryYAxis = {
      labelFormat: '${value}K'
    };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Enable Legend

You can use legend for the chart by setting the `visible` property to `true` in `legendSettings` object and by injecting the `LegendService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LegendService, LineSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LegendService, LineSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis' [legendSettings]='legendSettings'
[title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='month'
yName='sales' name='Sales'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public primaryYAxis?: Object;
  public legendSettings?: Object;
  public title?: string;
  ngOnInit(): void {
    // Legend for chart
    this.legendSettings = {
      visible: true
    }
    this.chartData = [
      { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
      { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
      { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
      { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
      { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
      { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
    ];
    this.primaryXAxis = {
      valueType: 'Category'
    };
    this.primaryYAxis = {
      labelFormat: '$ {value}K'
    };
    this.title = 'Sales Analysis';
  }
}
```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Add Data Label

You can add data labels to improve the readability of the chart.

This can be achieved by setting the visible property to true in the `dataLabel` object and by injecting `DataLabelService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LegendService, DataLabelService,
LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LegendService, DataLabelService,
LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis' [legendSettings]='legendSettings'
[title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='month'
yName='sales' name='Sales' [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public primaryYAxis?: Object;
  public legendSettings?: Object;
  public marker?: Object;
  public title?: string;
  ngOnInit(): void {
    // Data label for chart series
    this.marker = {
      dataLabel: {
        visible: true
      }
    };
    this.chartData = [
```

```

        { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
        { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
        { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
        { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
        { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
        { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
    ];
    this.primaryXAxis = {
        valueType: 'Category'
    };
    this.primaryYAxis = {
        labelFormat: '${value}K'
    };
    this.legendSettings = {
        visible: true
    };
    this.title = 'Sales Analysis';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Enable Tooltip

The tooltip is useful when you cannot display information by using the data labels due to space constraints. You can enable tooltip by setting the enable property as true in [tooltip](#) object and by injecting `TooltipService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LegendService, TooltipService, DataLabelService,
LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LegendService, TooltipService,
DataLabelService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis'
[legendSettings]='legendSettings' [tooltip]='tooltip' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='month'
yName='sales' name='Sales' [marker]='marker'></e-series>
    </e-series-collection>
  `
})

```



```

    </ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public primaryYAxis?: Object;
    public legendSettings?: Object;
    public tooltip?: Object;
    public title?: string;
    public marker?: Object;
    ngOnInit(): void {
      // Tooltip for chart
      this.tooltip = {
        enable: true
      }
      this.chartData = [
        { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
        { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
        { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
        { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
        { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
        { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
      ];
      this.primaryXAxis = {
        valueType: 'Category'
      };
      this.primaryYAxis = {
        labelFormat: '${value}K'
      };
      this.marker = {
        dataLabel: {
          visible: true
        }
      };
      this.legendSettings = {
        visible: true
      };
      this.title = 'Sales Analysis';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

You can also explore our [Angular Charts example](#) that shows various chart types and how to represent time-dependent data, showing trends in data at equal intervals.

<!-- markdownlint-disable MD036 -->

Working with data in Angular Chart component

Chart can visualise data bound from local or remote data.

Local Data

You can bind a simple JSON data to the chart using [dataSource](#) property in series. Now map the fields in JSON to [xName](#) and [yName](#) properties.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
  ColumnSeriesService, LineSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
  StackingColumnSeriesService, LegendService, TooltipService
  } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService,],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='month'
yName='sales' name='Sales'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  ngOnInit(): void {
    this.chartData = [
      { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
      { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
      { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
      { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
      { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
      { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
    ];
    this.primaryXAxis = {
      valueType: 'Category'
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Common Datasource

You can also bind a JSON data common to all series using [dataSource](#) property in chart.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
  ColumnSeriesService, LineSeriesService,
  ChartAnnotationService, RangeColumnSeriesService,
  StackingColumnSeriesService, LegendService, TooltipService
  } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService,],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[dataSource]='chartData'>
  <e-series-collection>
    <e-series type='Column' xName='month' yName='sales'
name='Sales'></e-series>
    <e-series type='Column' xName='month' yName='sales1'
name='Sales'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  ngOnInit(): void {
    this.chartData = [
      { month: 'Jan', sales: 35, sales1: 28 }, { month: 'Feb', sales: 28,
sales1: 35 },
      { month: 'Mar', sales: 34, sales1: 32 }, { month: 'Apr', sales: 32,
sales1: 34 },
      { month: 'May', sales: 40, sales1: 32 }, { month: 'Jun', sales: 32,
sales1: 40 },
      { month: 'Jul', sales: 35, sales1: 55 }, { month: 'Aug', sales: 55,
sales1: 35 },
      { month: 'Sep', sales: 38, sales1: 30 }, { month: 'Oct', sales: 30,
sales1: 38 },
      { month: 'Nov', sales: 25, sales1: 32 }, { month: 'Dec', sales: 32,
sales1: 25 }
    ];
    this.primaryXAxis = {
      valueType: 'Category'
    }
  }
}
```

```

    };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Lazy loading

Lazy loading allows you to load data for chart on demand. Chart will fire the scrollEnd event, in that we can get the minimum and maximum range of the axis, based on this, we can upload the data to chart.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
  ColumnSeriesService, LineSeriesService,
  ChartAnnotationService, RangeColumnSeriesService,
  StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { ChartComponent } from '@syncfusion/ej2-angular-charts';
import { Internationalization } from '@syncfusion/ej2-base';
import { NumericTextBoxComponent } from '@syncfusion/ej2-angular-inputs';
import { IScrollEventArgs } from '@syncfusion/ej2-charts';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService, ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart style='display:block;' #chart
[legendSettings]='legend' id='container' [primaryXAxis]='primaryXAxis'
[tooltip]='tooltip' [height]='height' [width]='width'
(scrollEnd)='scrollEnd($event)'
[primaryYAxis]='primaryYAxis' [crosshair]='crosshair'
[chartArea]='chartArea' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='data' [animation]='animation'
type='Line' xName='x' yName='y'>
    </e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  ngOnInit(): void {

```

```

    }
    public intl: Internationalization = new Internationalization();
    @ViewChild('point')
    private pointslength?: NumericTextBoxComponent;
    public value: number = 1000;
    public step: number = 100;
    public enabled: boolean = false;
    public format: string = 'n';
    public dropValue: string = 'Range';
    public minValue: Date = new Date(2009, 0, 1);
    public maxValue: Date = new Date(2014, 0, 1);
    public dropDownData: Object = [
        { value: 'Range' },
        { value: 'Points Length' }
    ];
    public fields: Object = { text: 'value', value: 'value' };
    public data: Object[] = this.GetNumericData(new Date(2009, 0, 1));
    @ViewChild('chart')
    public chart?: ChartComponent;
    // Initializing Primary X Axis
    public primaryXAxis: Object = {
        title: 'Day',
        valueType: 'DateTime',
        edgeLabelPlacement: 'Shift',
        skeleton: 'yMMM',
        skeletonType: 'Date',
        scrollbarSettings: {
            range: {
                minimum: new Date(2009, 0, 1),
                maximum: new Date(2014, 0, 1)
            },
            enable: true,
            pointsLength: 1000
        }
    };
    public height: string = '450';
    public width: string = '100%';
    //Initializing Primary Y Axis
    public primaryYAxis: Object = {
        title: 'Server Load',
        labelFormat: '{value}MB'
    };
    public tooltip: Object = {
        enable: true, shared: true,
        header : "<b>${point.x}</b>", format : "Server load :
        <b>${point.y}</b>"
    };
    public legend: Object = {
        visible: false
    };
    public title: string = 'Network Load';
    public animation: Object = { enable: false };
    public chartArea: Object = {
        border: {
            width: 0
        }
    };
};

```

```

crosshair: any;
    public scrollEnd(args: IScrollEventArgs | any): void {
        (this.chart as ChartComponent).series[0].dataSource =
this.GetNumericData(new Date(args.currentRange.maximum));
        (this.chart as ChartComponent).dataBind();
    };
    public GetNumericData(date: Date): {x: Date, y: number}[] {
        var series1 = [];
        var value = 30;
        for (var i = 0; i <= 60; i++) {
            if (Math.random() > .5) {
                value += (Math.random() * 10 - 5);
            }
            else {
                value -= (Math.random() * 10 - 5);
            }
            if (value < 0) {
                value = this.getRandomInt(20, 40);
            }
            date = new Date(date.setMinutes(date.getMinutes() + 1));
            var point = { x: date, y: Math.round(value) };
            series1.push(point);
        }
        return series1;
    }
    public getRandomInt(min: number, max: number) {
        return Math.floor(Math.random() * (max - min + 1)) + min;
    }
};

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Remote Data

You can also bind remote data to the chart using **DataManager**. The DataManager requires minimal information like webservice URL, adaptor and crossDomain to interact with service endpoint properly. Assign the instance of DataManager to the [dataSource](#) property in series and map the fields of data to [xName](#) and [yName](#) properties. You can also use the [query](#) property of the series to filter the data.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
ColumnSeriesService, LineSeriesService,
ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query } from '@syncfusion/ej2-data';

```

```

@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService, ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis'>
    <e-series-collection>
      <e-series [dataSource]='dataManager' type='Column'
[query]='query' xName='CustomerID' yName='Freight' name='Sales'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public primaryYAxis?: Object;
  public dataManager: DataManager = new DataManager({
    url: 'https://ej2services.syncfusion.com/production/web-
services/api/Orders'
  });
  public query: Query = new Query().take(5).where('Estimate', 'lessThan',
3, false);
  ngOnInit(): void {
    this.primaryXAxis = {
      rangePadding: 'Additional',
      valueType: 'Category',
      title: 'Assignee'
    };
    this.primaryYAxis = {
      title: 'Estimate'
    };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Empty points

The Data points that uses the **null** or **undefined** as value are considered as empty points. Empty data points are ignored and not plotted in the Chart.

When the data is provided by using the points property, By using **emptyPointSettings** property in series, you can customize the empty point. Default **mode** of the empty point is **Gap**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
ColumnSeriesService, LineSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService
    } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
imports: [
    ChartModule
],
providers: [ CategoryService, DateTimeService, ScrollBarService,
LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService,],
standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='month'
yName='sales' name='Sales' [emptyPointSettings]='emptySeries1'></e-series>
    </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public marker?: Object;
    public emptySeries1?: Object;
    public emptySeries2?: Object;
    ngOnInit(): void {
        this.chartData = [
            { month: 'Jan', sales: 35 }, { month: 'Feb', sales: null },
            { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
            { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
            { month: 'Jul', sales: undefined }, { month: 'Aug', sales: 55 },
            { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
            { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
        ];
        this.primaryXAxis = {
            valueType: 'Category'
        }
        this.marker = {
            visible : true
        }
        this.emptySeries1 = {
            mode: 'Gap'
        }
        this.emptySeries2 = {
            mode: 'Average'
        }
    }
}

```


MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customizing empty point

Specific color for empty point can be set by **fill** property in **emptyPointSettings**. Border for a empty point can be set by **border** property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateFormatOptions } from '@syncfusion/ej2-base'
import { CategoryService, DateTimeService, ScrollBarService,
    ColumnSeriesService, LineSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService, ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='month'
yName='sales' name='Sales' [emptyPointSettings]='emptySeries1'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public marker?: Object;
  public emptySeries1?: Object;
  public emptySeries2?: Object;
  ngOnInit(): void {
    this.chartData = [
      { month: 'Jan', sales: 35 }, { month: 'Feb', sales: null },
      { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
      { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
      { month: 'Jul', sales: undefined }, { month: 'Aug', sales: 55 },
      { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
```

```

        { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
    ];
    this.primaryXAxis = {
        valueType: 'Category'
    }
    this.marker = {
        visible : true
    }
    this.emptySeries1 = {
        mode: 'Average',
        fill: 'red',
        border: { width: 2, color: 'violet' }
    }
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Chart dimensions in Angular Chart component

Size for Container

Chart can render to its container size. You can set the size via inline or CSS as demonstrated below.

,

```

<div style="width:650px; height:350px;">
<ejs-chart id="chart-container"></ejs-chart>
</div>

```

,

`javascript

```

import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-container',
  template:
    `<div style="width:650px; height:350px;">
    <ejs-chart id="chart-container"></ejs-chart>
    </div>`
})
export class AppComponent {
  constructor(){

```

```

/*
*/
}
}
`

```

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<div style="width:650px; height:350px;">
    <ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'>
      <e-series-collection>
        <e-series [dataSource]='chartData' type='Line' xName='month'
yName='sales' name='Sales'></e-series>
      </e-series-collection>
    </ejs-chart>
  </div>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  ngOnInit(): void {
    this.chartData = [
      { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
      { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
      { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
      { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
      { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
      { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
    ];
    this.primaryXAxis = {
      valueType: 'Category'
    };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Size for Chart

You can also set size for chart directly through [width](#) and [height](#) properties.

<!-- markdownlint-disable MD036 -->

In Pixel

<!-- markdownlint-disable MD036 -->

You can set the size of chart in pixel as demonstrated below.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
width='650' height='350'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Line' xName='month'
yName='sales' name='Sales'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  ngOnInit(): void {
    this.chartData = [
      { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
      { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
      { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
      { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
      { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
      { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
    ];
    this.primaryXAxis = {
      valueType: 'Category'
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

In Percentage

By setting value in percentage, chart gets its dimension with respect to its container. For example, when the height is '50%', chart renders to half of the container height.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
width='80%' height='90%'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='month'
yName='sales' name='Sales'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  ngOnInit(): void {
    this.chartData = [
      { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
      { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
      { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
      { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
      { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
      { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
    ];
    this.primaryXAxis = {
      valueType: 'Category'
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Note: When you do not specify the size, it takes [Link to the Video](#) as the height and window size as its width.

Category axis in Angular Chart component

<!-- markdownlint-disable MD036 -->

Category axis are used to represent, the string values instead of numbers.

To know about category axis, you can check on this video:

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = [
      { country: "USA", gold: 50 },
      { country: "China", gold: 40 },
      { country: "Japan", gold: 70 },
      { country: "Australia", gold: 60 },
      { country: "France", gold: 50 },
      { country: "Germany", gold: 40 },
      { country: "Italy", gold: 40 },
      { country: "Sweden", gold: 30, silver: 25 }
    ];
    this.primaryXAxis = {
      valueType: 'Category',
      title: 'Countries'
    };
  }
}
```

```

        this.primaryYAxis = {
            minimum: 0, maximum: 80,
            interval: 20, title: 'Medals'
        };
        this.title = 'Olympic Medals';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note: To use category axis, we need to inject `CategoryService` into the `@NgModule.providers` and set the `valueType` of axis to `Category`.

<!-- markdownlint-disable MD036 -->

Labels Placement

<!-- markdownlint-disable MD036 -->

By default, category labels are placed between the ticks in an axis, this can also be placed on ticks using [labelPlacement](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { categoryData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;

```

```

public chartData?: Object[];
public title?: string;
primaryYAxis: any;
ngOnInit(): void {
    this.chartData = categoryData;
    this.primaryXAxis = {
        valueType: 'Category',
        title: 'Countries',
        labelPlacement: 'OnTicks'
    };
    this.title = 'Olympic Medals';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Range

Range of the category axis can be customized using [minimum](#), [maximum](#) and [interval](#) property of the axis.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { categoryData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[tittle]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
}

```



```

public title?: string;
public primaryYAxis?: Object;
ngOnInit(): void {
    this.chartData = categoryData;
    this.primaryXAxis = {
        valueType: 'Category',
        minimum: 1,
        maximum: 5,
        interval: 2,
        title: 'Countries'
    };
    this.title = 'Olympic Medals';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Indexed category axis

Category axis can be rendered based on the index values of data source. This can be achieved by defining the `isIndexed` property to `true` in the axis.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChartModule } from '@syncfusion/ej2-angular-charts';
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
import { categoryData } from './datasource';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='chartData1' type='Column' xName='x'
yName='y'></e-series>
        <e-series [dataSource]='chartData2' type='Column' xName='x'
yName='y'></e-series>
    </e-series-collection>
    </ejs-chart>`
})

```

```
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData1?: Object[];
  public chartData2?: Object[];
  public title?: string;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData1 = [{ x: 'Myanmar', y: 7.3 }, { x: 'India', y: 7.9 }, { x: 'Bangladesh', y: 6.8 }, { x: 'Cambodia', y: 7.0 }, { x: 'China', y: 6.9 }];
    this.chartData2 = [{ x: 'Poland', y: 2.7 }, { x: 'Australia', y: 2.5 }, { x: 'Singapore', y: 2.0 }, { x: 'Canada', y: 1.4 }, { x: 'Germany', y: 1.8 }];
    this.primaryXAxis = {
      valueType: 'Category',
      isIndexed: true
    };
    this.title = 'Olympic Medals';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

<!-- markdownlint-disable MD036 -->

Numeric axis in Angular Chart component

You can use [numeric axis](#) to represent numeric values of data in chart. By default, the `valueType` of an axis is [Link to the Video](#).

To know about numeric axis, you can check on this video:

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChartModule } from '@syncfusion/ej2-angular-charts';
import { ColumnSeriesService, AreaSeriesService } from '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ ColumnSeriesService, AreaSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Area' xName='x'
yName='y' name='England'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData1?: Object[];
  public chartData2?: Object[];
  public title?: string;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData1 = [{ x: 'Myanmar', y: 7.3 }, { x: 'India', y: 7.9 }, { x: 'Bangladesh', y: 6.8 }, { x: 'Cambodia', y: 7.0 }, { x: 'China', y: 6.9 }];
    this.chartData2 = [{ x: 'Poland', y: 2.7 }, { x: 'Australia', y: 2.5 }, { x: 'Singapore', y: 2.0 }, { x: 'Canada', y: 1.4 }, { x: 'Germany', y: 1.8 }];
    this.primaryXAxis = {
      valueType: 'Category',
      isIndexed: true
    };
    this.title = 'Olympic Medals';
  }
}
```

```

        </e-series-collection>
    </ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public primaryYAxis?: Object;
        ngOnInit(): void {
            this.chartData = [
                { x: 1, y: 7 }, { x: 2, y: 1 }, { x: 3, y: 1 }, { x: 4, y: 14 }, { x: 5,
y: 1 }, { x: 6, y: 10 },
                { x: 7, y: 8 }, { x: 8, y: 6 }, { x: 9, y: 10 }, { x: 10, y: 10 }, { x:
11, y: 16 }, { x: 12, y: 6 },
                { x: 13, y: 14 }, { x: 14, y: 7 }, { x: 15, y: 5 }, { x: 16, y: 2 }, {
x: 17, y: 14 }, { x: 18, y: 7 },
                { x: 19, y: 7 }, { x: 20, y: 10 }];
            this.title = 'England - Run Rate';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Range

Range for an axis, will be calculated automatically based on the provided data, you can also customize the range of the axis using [minimum](#), [maximum](#) and [interval](#) property of the axis.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ColumnSeriesService, AreaSeriesService } from '@syncfusion/ej2-
angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ ColumnSeriesService, AreaSeriesService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Area' xName='x'
yName='y' name='England'></e-series>
    </e-series-collection>
</ejs-chart>`
})

```

```

    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public primaryYAxis?: Object;
        ngOnInit(): void {
            this.primaryXAxis = {
                valueType: 'Double',
                minimum: 1,
                maximum: 20,
                interval: 5,
                title: 'Overs'
            };
            this.chartData = chartData;
            this.title = 'England - Run Rate';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Range Padding

Padding can be applied to the minimum and maximum extremes of the axis range by using the [rangePadding](#) property. Numeric axis supports following types of padding.

- None
- Round
- Additional
- Normal
- Auto

Numeric - None

When the [rangePadding](#) is set to **None**, minimum and maximum of an axis is based on the data.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChartModule } from '@syncfusion/ej2-angular-charts';
import { ColumnSeriesService, AreaSeriesService } from '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ ColumnSeriesService, AreaSeriesService],
})

```

```

standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Area' xName='x'
yName='y' name='England'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = chartData;
    this.primaryXAxis = {
      valueType: 'Double',
      title: 'Overs'
    };
    this.primaryYAxis = {
      title: 'Runs',
      valueType: 'Double',
      rangePadding: 'None'
    };
    this.title = 'England - Run Rate';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Numeric - Round

When the [rangePadding](#) is set to **Round**, minimum and maximum will be rounded to the nearest possible value divisible by interval. For example, when the minimum is 3.5 and the interval is 1, then the minimum will be rounded to 3.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ColumnSeriesService, AreaSeriesService } from '@syncfusion/ej2-
angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],

```

```

providers: [ ColumnSeriesService, AreaSeriesService],
standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Area' xName='x'
yName='y' name='England'></e-series>
    </e-series-collection>
  </ejs-chart>`
}))
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = chartData;
    this.primaryXAxis = {
      valueType: 'Double',
      title: 'Overs'
    };
    this.primaryYAxis = {
      title: 'Runs',
      valueType: 'Double',
      rangePadding: 'Round'
    };
    this.title = 'England - Run Rate';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Numeric - Additional

When the [rangePadding](#) is set to **Additional**, interval of an axis will be padded to the minimum and maximum of the axis.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ColumnSeriesService, AreaSeriesService } from '@syncfusion/ej2-
angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],

```

```

providers: [ ColumnSeriesService, AreaSeriesService],
standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Area' xName='x'
yName='y' name='England'></e-series>
    </e-series-collection>
  </ejs-chart>`
}))
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = chartData;
    this.primaryXAxis = {
      valueType: 'Double',
      title: 'Overs'
    };
    this.primaryYAxis = {
      title: 'Runs',
      valueType: 'Double',
      rangePadding: 'Additional'
    };
    this.title = 'England - Run Rate';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Numeric - Normal

When the [rangePadding](#) is set to **Normal**, padding is applied to the axis based on default range calculation.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ColumnSeriesService, AreaSeriesService } from '@syncfusion/ej2-
angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],

```

```

providers: [ ColumnSeriesService, AreaSeriesService],
standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Area' xName='x'
yName='y' name='England'></e-series>
    </e-series-collection>
  </ejs-chart>`
}))
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = chartData;
    this.primaryXAxis = {
      valueType: 'Double',
      title: 'Overs'
    };
    this.primaryYAxis = {
      title: 'Runs',
      valueType: 'Double',
      rangePadding: 'Normal'
    };
    this.title = 'England - Run Rate';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Numeric - Auto

When the [rangePadding](#) is set to **Auto**, horizontal numeric axis takes None as padding calculation, while the vertical numeric axis takes Normal as padding calculation.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ColumnSeriesService, AreaSeriesService } from '@syncfusion/ej2-
angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],

```



```

providers: [ ColumnSeriesService, AreaSeriesService],
standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Area' xName='x'
yName='y' name='England'></e-series>
    </e-series-collection>
  </ejs-chart>`
}))
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = chartData;
    this.primaryXAxis = {
      valueType: 'Double',
      title: 'Overs'
    };
    this.primaryYAxis = {
      title: 'Runs',
      valueType: 'Double',
      rangePadding: 'Auto'
    };
    this.title = 'England - Run Rate';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Label Format

Numeric Label Format

Numeric labels can be formatted by using the [labelFormat](#) property.

Numeric labels supports all globalize format.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ColumnSeriesService, AreaSeriesService } from '@syncfusion/ej2-
angular-charts'
import { Component, OnInit } from '@angular/core';
import { formatData } from './datasource';
@Component({
  imports: [

```

```

    ChartModule
  ],
  providers: [ ColumnSeriesService, AreaSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='formatData' type='Area' xName='x'
yName='y' name='Product X' opacity=0.6></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public formatData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.formatData = formatData;
    this.primaryXAxis = {
      title: 'Year',
      edgeLabelPlacement: 'Shift'
    };
    this.primaryYAxis = {
      title: 'Sales Amount in Millions',
      labelFormat: 'c'
    };
    this.title = 'Average Sales Comparison';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

The following table describes the result of applying some commonly used label formats on numeric values.

<!-- markdownlint-disable MD033 -->

Label Value	Label Format property value	Result	Description
1000	n1	1000.0	The Number is rounded to 1 decimal place
1000	n2	1000.00	The Number is rounded to 2 decimal place
1000	n3	1000.000	The Number is rounded to 3 decimal place
0.01	p1	1.0%	The Number is converted to percentage with 1 decimal place

0.01	p2	1.00%	The Number is converted to percentage with 2 decimal place
0.01	p3	1.000%	The Number is converted to percentage with 3 decimal place
1000	c1	\$1000.0	The Currency symbol is appended to number and number is rounded to 1 decimal place
1000	c2	\$1000.00	The Currency symbol is appended to number and number is rounded to 2 decimal place

GroupingSeparator

To separate groups of thousands, use [Link to the Video](#) property in chart.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ColumnSeriesService, AreaSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { groupingData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ ColumnSeriesService, AreaSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
  useGroupingSeparator=true>
    <e-series-collection>
      <e-series [dataSource]='groupingData' type='Column' xName='x'
yName='y' name='Product X' opacity=0.6></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public groupingData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.groupingData = groupingData;
    this.primaryXAxis = {
      title: 'Quantity',
      edgeLabelPlacement: 'Shift'
    };
    this.primaryYAxis = {
      title: 'Sales Amount in Millions',
    };
    this.title = 'Average Sales Comparison';
  }
}
```

```
}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Custom Label Format

Axis also supports custom label format using placeholder like {value}°C, in which the value represent the axis label e.g 20°C.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ColumnSeriesService, AreaSeriesService } from '@syncfusion/ej2-
angular-charts'
import { Component, OnInit } from '@angular/core';
import { formatData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ ColumnSeriesService, AreaSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='formatData' type='Area' xName='x'
yName='y' name='Product X' opacity=0.6></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public formatData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.formatData = formatData;
    this.primaryXAxis = {
      title: 'Year',
      edgeLabelPlacement: 'Shift'
    };
    this.primaryYAxis = {
      title: 'Sales Amount in Millions',
      labelFormat: '${value}k'
    };
    this.title = 'Average Sales Comparison';
  }
}
```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

```
<!-- markdownlint-disable MD036 -->
```

Date time axis in Angular Chart component**DateTime Axis**

Date time axis uses date time scale and displays the date time values as axis labels in the specified format.

To know about date time axis, you can check on this video:

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Sales'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = [
      { x: new Date(2000, 6, 11), y: 10 }, { x: new Date(2002, 3,
7), y: 30 },
      { x: new Date(2004, 3, 6), y: 15 }, { x: new Date(2006, 3,
30), y: 65 },
      { x: new Date(2008, 3, 8), y: 90 }, { x: new Date(2010, 3,
8), y: 85 }
```

```

    ];
    this.primaryXAxis = {
      valueType: 'DateTime',
      title: 'Sales Across Years',
      labelFormat: 'yMMM'
    };
    this.primaryYAxis = {
      title: 'Sales Amount in millions(USD)'
    };
    this.title = 'Average Sales Comparison';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note: To use datetime axis, we need to inject `DateTimeService` into the `@NgModule.providers` and set the `valueType` of axis to `DateTime`.

DateTimeCategory Axis

Date-time category axis is used to display the date-time values with non-linear intervals. For example, the business days alone have been depicted in a week here.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { data } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Sales'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];

```

```

public title?: string;
public primaryYAxis?: Object;
ngOnInit(): void {
    this.chartData = data;
    this.primaryXAxis = {
        valueType: 'DateTimeCategory'
    };
    this.primaryYAxis = {
        title: 'Sales Amount in millions(USD)'
    };
    this.title = 'Average Sales Comparison';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note: To use datetime axis, we need to inject `DateTimeCategoryService` into the `@NgModule.providers` and set the `valueType` of axis to `DateTimeCategory`.

Range

Range for an axis, will be calculated automatically based on the provided data, you can also customize the range of the axis using `minimum`, `maximum` and `interval` property of the axis.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { dateData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Sales'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;

```

```

public chartData?: Object[];
public title?: string;
primaryYAxis:any;
ngOnInit(): void {
    this.chartData = dateData;
    this.primaryXAxis = {
        valueType: 'DateTime',
        title: 'Sales Across Years',
        labelFormat: 'yMMM',
        minimum: new Date(2000, 6, 1),
        maximum: new Date(2010, 6, 1)
    };
    this.title = 'Average Sales Comparison';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Interval Customization

Date time intervals can be customized by using the [interval](#) and [intervalType](#) properties of the axis.

For example, when you set interval as 2 and intervalType as years, it considers 2 years as interval.

DateTime axis supports following interval types,

- Auto
- Years
- Months
- Days
- Hours
- Minutes
- Seconds

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { dateData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService],
  standalone: true,

```



```

    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Sales'></e-series>
    </e-series-collection>
</ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    primaryYAxis: any;
    ngOnInit(): void {
      this.chartData = dateData;
      this.primaryXAxis = {
        valueType: 'DateTime',
        intervalType: 'Years'
      };
      this.title = 'Average Sales Comparison';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Applying Padding to the Range

Padding can be applied to the minimum and maximum extremes of the range by using the [rangePadding](#) property. Date time axis supports the following types of padding,

- None
- Round
- Additional

DateTime - None

When the [rangePadding](#) is set to **None**, minimum and maximum of the axis is based on the data.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { dateData } from './datasource';
@Component({
  imports: [

```

```

        ChartModule
    ],
    providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
        StripLineService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
        <e-series-collection>
            <e-series [dataSource]='chartData' type='Line' xName='x'
    yName='y' name='Sales'></e-series>
        </e-series-collection>
    </ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        primaryYAxis: any;
        ngOnInit(): void {
            this.chartData = dateData;
            this.primaryXAxis = {
                valueType: 'DateTime',
                title: 'Sales Across Years',
                labelFormat: 'yMMM',
                rangePadding: 'None'
            };
            this.title = 'Average Sales Comparison';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

DateTime - Round

When the [rangePadding](#) is set to **Round**, minimum and maximum will be rounded to the nearest possible value divisible by interval. For example, when the minimum is 15th Jan, interval is 1 and the interval type is 'month', then the axis minimum will be Jan 1st.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
    StripLineService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { dateData } from './datasource';
@Component({
    imports: [
        ChartModule
    ]
})

```

```

    ],
    providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
    StripLineService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
      <e-series-collection>
        <e-series [dataSource]='chartData' type='Line' xName='x'
        yName='y' name='Sales'></e-series>
      </e-series-collection>
    </ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    primaryYAxis: any;
    ngOnInit(): void {
      this.chartData = dateData;
      this.primaryXAxis = {
        valueType: 'DateTime',
        title: 'Sales Across Years',
        labelFormat: 'yMMM',
        rangePadding: 'Round'
      };
      this.title = 'Average Sales Comparison';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

DateTime - Additional

When the [rangePadding](#) is set to **Additional**, interval of an axis will be padded to the minimum and maximum of the axis.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { dateData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],

```

```

providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService],
standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Sales'></e-series>
    </e-series-collection>
  </ejs-chart>`
}))
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = dateData;
    this.primaryXAxis = {
      valueType: 'DateTime',
      title: 'Sales Across Years',
      labelFormat: 'yMMM',
      rangePadding: 'Additional'
    };
    this.title = 'Average Sales Comparison';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Label Format

You can format and parse the date to all globalize format using [Link to the Video](#) property in an axis.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { dateData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService],
  standalone: true,
  selector: 'app-container',

```

```

    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Sales'></e-series>
    </e-series-collection>
</ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    ngOnInit(): void {
      this.chartData = dateData;
      this.primaryXAxis = {
        valueType: 'DateTime',
        title: 'Sales Across Years',
        labelFormat: 'yMd'
      };
      this.primaryYAxis = {
        title: 'Sales Amount in millions(USD)'
      };
      this.title = 'Average Sales Comparison';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

The following table describes the result of applying some common date time formats to the labelFormat property

<!-- markdownlint-disable MD033 -->

Label Value	Label Format Property Value	Result	Description
new Date(2000, 03, 10)	EEEE	Monday	The Date is displayed in day format
new Date(2000, 03, 10)	yMd	04/10/2000	The Date is displayed in month/date/year format
new Date(2000, 03, 10)	MMM	Apr	The Shorthand month for the date is displayed
new Date(2000, 03, 10)	hm	12:00 AM	Time of the date value is displayed as label

new Date(2000, 03, 10)	hms	12:00:00 AM	The Label is displayed in hours:minutes:seconds format
------------------------	-----	-------------	--

<!-- markdownlint-disable MD033 -->

Logarithmic axis in Angular Chart component

<!-- markdownlint-disable MD033 -->

Logarithmic axis uses logarithmic scale and it is very useful in visualizing data, when it has numeric values in both lower order of magnitude (eg: 10^{-6}) and higher order of magnitude (eg: 10^6).

To know about logarithmic axis, you can check on this video:

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { LogarithmicService, DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ LogarithmicService, LineSeriesService, DateTimeService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Product X'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = [
      { x: new Date(1995, 0, 1), y: 80 }, { x: new Date(1996, 0,
1), y: 200 },
      { x: new Date(1997, 0, 1), y: 400 }, { x: new Date(1998, 0,
1), y: 600 },
      { x: new Date(1999, 0, 1), y: 700 }, { x: new Date(2000, 0,
1), y: 1400 },
      { x: new Date(2001, 0, 1), y: 2000 }, { x: new Date(2002, 0,
1), y: 4000 },
      { x: new Date(2003, 0, 1), y: 6000 }, { x: new Date(2004, 0,
1), y: 8000 },
      { x: new Date(2005, 0, 1), y: 11000 }
    ]
  }
}
```

```

    ];
    this.primaryXAxis = {
      valueType: 'DateTime',
      title: 'Years',
      labelFormat: 'y'
    };
    this.primaryYAxis = {
      valueType: 'Logarithmic',
      title: 'Profit'
    };
    this.title = 'Product X Growth [1995-2005]';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note: To use log axis, we need to inject `LogarithmicService` into the `@NgModule.providers` and set the [valueType](#) of axis to `Logarithmic`.

Range

Range of an axis, will be calculated automatically based on the provided data, you can also customize the range of the axis using [minimum](#), [maximum](#) and [interval](#) property of the axis.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { LogarithmicService, DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { logData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ LogarithmicService, LineSeriesService, DateTimeService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Product X'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];

```

```

public title?: string;
public primaryYAxis?: Object;
ngOnInit(): void {
    this.chartData = logData;
    this.primaryXAxis = {
        valueType: 'DateTime',
        title: 'Years',
        labelFormat: 'y'
    };
    this.primaryYAxis = {
        valueType: 'Logarithmic',
        title: 'Profit',
        minimum: 100,
        maximum: 10000
    };
    this.title = 'Product X Growth [1995-2005]';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Logarithmic Base

Logarithmic base can be customized by using the [logBase](#) property of the axis.

For example when the logBase is 5, the axis values follows 5^{-2} , 5^{-1} , 5^0 ,

5^1 , 5^2 etc.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { LogarithmicService, DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { logData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ LogarithmicService, LineSeriesService, DateTimeService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Product X'></e-series>
    </e-series-collection>
`
})

```



```

    </ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    ngOnInit(): void {
      this.chartData = logData;
      this.primaryXAxis = {
        valueType: 'DateTime',
        title: 'Years',
        labelFormat: 'Y'
      };
      this.primaryYAxis = {
        valueType: 'Logarithmic',
        title: 'Profit',
        logBase: 2
      };
      this.title = 'Product X Growth [1995-2005]';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Logarithmic Interval

Logarithmic axis interval can be customized by using the [interval](#) property of the axis. When the logarithmic base is 10 and logarithmic interval is 2, then the axis labels are placed at an interval of 10^2 . The default value of the interval is 1.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { LogarithmicService, DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { logData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ LogarithmicService, LineSeriesService, DateTimeService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>

```

```

        <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Product X'></e-series>
    </e-series-collection>
</ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public primaryYAxis?: Object;
        ngOnInit(): void {
            this.chartData = logData;
            this.primaryXAxis = {
                valueType: 'DateTime',
                title: 'Years',
                labelFormat: 'y'
            };
            this.primaryYAxis = {
                valueType: 'Logarithmic',
                title: 'Profit',
                interval: 2
            };
            this.title = 'Product X Growth [1995-2005]';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Axis labels in Angular Chart component

Smart Axis Labels

When the axis labels overlap with each other, you can use [labelIntersectAction](#) property in the axis, to place them smartly.

When setting `labelIntersectAction` as `Hide`

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService, LineSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ CategoryService, ColumnSeriesService, LineSeriesService],
    standalone: true,
    selector: 'app-container',

```

```

    template: `<ejs-chart id="chart-container" width='450px'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Internet'></e-series>>
    </e-series-collection>
</ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    ngOnInit(): void {
      this.chartData = [
        { x: "South Korea", y: 39.4 }, { x: "India", y: 61.3 }, { x:
"Pakistan", y: 20.4 },
        { x: "Germany", y: 65.1 }, { x: "Australia", y: 15.8 }, { x:
"Italy", y: 29.2 },
        { x: "France", y: 44.6 }, { x: "Saudi Arabia", y: 9.7 }, {
x: "Russia", y: 40.8 },
        { x: "Mexico", y: 31 }, { x: "Brazil", y: 75.9 }, { x:
"China", y: 51.4 }
      ];
      this.primaryXAxis = {
        title: 'Countries',
        valueType: 'Category',
        labelIntersectAction: 'Hide'
      };
      this.primaryYAxis = {
        minimum: 0, maximum: 80, interval: 10,
        title: 'People(in millions)'
      };
      this.title = 'Internet Users';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

When setting `labelIntersectAction` as `Rotate45`

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService, LineSeriesService } from
'@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { labelData } from './datasource';
@Component({

```

```

imports: [
    ChartModule
],
providers: [ CategoryService, ColumnSeriesService, LineSeriesService],
standalone: true,
selector: 'app-container',
template: `<ejs-chart id="chart-container" width='450px'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Internet'></e-series>>
    </e-series-collection>
</ejs-chart>`
))
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    ngOnInit(): void {
        this.chartData = labelData;
        this.primaryXAxis = {
            title: 'Countries',
            valueType: 'Category',
            labelIntersectAction: 'Rotate45'
        };
        this.primaryYAxis = {
            minimum: 0, maximum: 80, interval: 10,
            title: 'People(in millions)'
        };
        this.title = 'Internet Users';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

When setting `labelIntersectAction` as `Rotate90`

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService, LineSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { labelData } from './datasource';
@Component({
    imports: [
        ChartModule
    ],

```

```

providers: [ CategoryService, ColumnSeriesService, LineSeriesService],
standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" width='450px'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Internet'></e-series>>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = labelData;
    this.primaryXAxis = {
      title: 'Countries',
      valueType: 'Category',
      labelIntersectAction: 'Rotate90'
    };
    this.primaryYAxis = {
      minimum: 0, maximum: 80, interval: 10,
      title: 'People(in millions)'
    };
    this.title = 'Internet Users';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Axis Labels Positioning

By default, the axis labels can be placed at **outside** the axis line and this also can be placed at **inside** the axis line using the **labelPosition** property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { categoryData } from './datasource';
@Component({
  imports: [
    ChartModule

```

```

    ],
    providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
    LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
    SelectionService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
      <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='country'
        yName='gold' name='Gold' ></e-series>
      </e-series-collection>
    </ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis: any;

    ngOnInit(): void {
      this.chartData = categoryData;
      this.primaryXAxis = {
        valueType: 'Category',
        title: 'Countries',
        labelPosition: 'Inside'
      };
      this.title = 'Olympic Medals';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Multilevel Labels

Any number of levels of labels can be added to an axis using the `multiLevelLabels` property. This property can be configured using the following properties:

- Categories
- Overflow
- Alignment
- Text style
- Border

Note: To use multilevel label feature, we need to inject `MultiLevelLabel` into the `@NgModule.providers`.

Categories

Using the categories property, you can configure the `start`, `end`, `text`, and `maximumTextWidth` of multilevel labels.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { categoryData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public legendSettings: Object = { visible: false };
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = categoryData;
    this.primaryXAxis = {
      valueType: 'Category',
      multiLevelLabels: [{ categories: [
        {
          //Start and end values of the multi-level labels
          accepts number, date and string values
          start: -0.5,
          end: 3.5,
          //Multi-level label's text.
          text: 'Half Yearly 1',
        },
        { start: 3.5, end: 7.5, text: 'Half Yearly 2' },
      ]}],
    };
    this.title = 'Olympic Medals';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Overflow

Using the **overflow** property, you can **trim** or **wrap** the multilevel labels.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { categoryData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public legendSettings: Object = { visible: false };
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = categoryData;
    this.primaryXAxis = {
      valueType: 'Category',
      multiLevelLabels: [{
        categories: [{ start: -0.5, end: 3.5, text: 'Half Yearly 1',
maximumTextWidth: 50 },
          { start: 3.5, end: 7.5, text: 'Half Yearly 2',
maximumTextWidth: 50 } ] ],
      overflow: 'Trim'
    ]
  }
}
```



```

    };
    this.title = 'Olympic Medals';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Alignment

The **alignment** property provides option to position the multilevel labels at **far**, **center**, or **near**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { categoryData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public legendSettings: Object = { visible: false };
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = categoryData;
    this.primaryXAxis = {
      valueType: 'Category',
      multiLevelLabels: [{
        categories: [{ start: -0.5, end: 3.5, text: 'Half Yearly 1' },
          { start: 3.5, end: 7.5, text: 'Half Yearly 2' } ]],

```

```

        alignment : 'Far'
    }
    };
    this.title = 'Olympic Medals';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Text customization

The `textStyle` property of multilevel labels provides options to customize the `size`, `color`, `fontFamily`, `fontWeight`, `fontStyle`, `opacity`, `textAlignment` and `textOverflow`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { categoryData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public legendSettings: Object = { visible: false };
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = categoryData;
    this.primaryXAxis = {

```

```

        valueType: 'Category',
        multiLevelLabels:[{
            categories: [{start: -0.5, end: 3.5, text: 'Half Yearly 1' },
                { start: 3.5, end: 7.5, text: 'Half Yearly 2' }],
            textStyle:{size:'18px', color:'Red'}
        }]
    };
    this.title = 'Olympic Medals';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Border customization

Using the **border** property, you can customize the **width**, **color**, and **type**. The **type** of border are **Rectangle**, **Brace**, **WithoutBorder**, **WithoutTopBorder**, **WithoutTopandBottomBorder** and **CurlyBrace**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { categoryData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
}

```

```

public title?: string;
public legendSettings: Object = { visible: false};
primaryYAxis: any;
ngOnInit(): void {
    this.chartData = categoryData;
    this.primaryXAxis = {
        valueType: 'Category',
        multiLevelLabels:[{
            categories: [{start: -0.5, end: 3.5, text: 'Half Yearly 1' },
                { start: 3.5, end: 7.5, text: 'Half Yearly 2' }]],
            border:{type:'Brace', color:'Blue', width: 2},
        }]
    };
    this.title = 'Olympic Medals';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Edge Label Placement

Labels with long text at the edges of an axis may appear partially in the chart. To avoid this, use [edgeLabelPlacement](#) property in axis, which moves the label inside the chart area for better appearance or hides it.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { dateData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Sales'></e-series>
    </e-series-collection>
  </ejs-chart>`
})

```

```
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    public legendSettings: Object = { visible: false };
    ngOnInit(): void {
        this.chartData = dateData;
        this.primaryXAxis = {
            valueType: 'DateTime',
            title: 'Sales Across Years',
            labelFormat: 'yMMM',
            edgeLabelPlacement: 'Shift',
            minimum: new Date(2000, 6, 1),
            maximum: new Date(2010, 6, 1)
        };
        this.primaryYAxis = {
            title: 'Sales Amount in millions(USD)'
        };
        this.title = 'Average Sales Comparison';
    }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Trim using maximum label width

You can trim the label using [enableTrim](#) property and width of the labels can also be customized using [maximumLabelWidth](#) property in the axis, the value maximum label width is 34 by default.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { categoryData } from './datasource';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[legendSettings]='legendSettings'>`
```

```

        <e-series-collection>
            <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' ></e-series>
        </e-series-collection>
    </ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public primaryYAxis?: Object;
        public legendSettings: Object = { visible: false };
        ngOnInit(): void {
            this.chartData = [
                { x: "South Korea", y: 39.4 }, { x: "India", y: 61.3 }, { x: "Pakistan",
y: 20.4 },
                { x: "Germany", y: 65.1 }, { x: "Australia", y: 15.8 }, { x: "Italy", y:
29.2 },
                { x: "United Kingdom", y: 44.6 }, { x: "Saudi Arabia", y: 9.7 }, { x:
"Russia", y: 40.8 },
                { x: "Mexico", y: 31 }, { x: "Brazil", y: 75.9 }, { x: "China", y: 51.4
            }];
            this.primaryXAxis = {
                valueType: 'Category',
                enableTrim: true,
                maximumLabelWidth: '34',
            };
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Labels Customization

The [labelStyle](#) property of an axis provides options to customize the color, font-family, font-size and [Link to the Video](#) of the axis labels.

To know more about labels customization, you can check on this video:

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { categoryData } from './datasource';
@Component({
    imports: [

```

```

    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
    LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
    SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
    [legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
        yName='gold' name='Gold' ></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
        yName='silver' name='Silver'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
        yName='bronze' name='Bronze' ></e-series>
    </e-series-collection>
  </ejs-chart>`
  })
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public legendSettings: Object = { visible: false };
  ngOnInit(): void {
    this.chartData = categoryData;
    this.primaryXAxis = {
      valueType: 'Category',
      title: 'Countries'
    };
    this.primaryYAxis = {
      minimum: 0, maximum: 80,
      interval: 20, title: 'Medals',
      labelFormat: '${value}K',
      titleStyle: {
        size: '16px', color: 'grey',
        fontFamily: 'Segoe UI', fontWeight: 'bold'
      },
      labelStyle: {
        size: '14px', color: 'blue',
        fontFamily: 'Segoe UI', fontWeight: 'bold'
      }
    };
    this.title = 'Olympic Medals';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Label Rotation

The axis labels can be rotated based on the `labelRotation` property in `primaryXAxis`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { categoryData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
    <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver'></e-series>
    <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' ></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public legendSettings: Object = { visible: false };
  ngOnInit(): void {
    this.chartData = categoryData;
    this.primaryXAxis = {
      valueType: 'Category', labelRotation: 90,
      title: 'Countries'
    };
    this.primaryYAxis = {
      minimum: 0, maximum: 80,
      interval: 20, title: 'Medals',
      labelFormat: '${value}K',
      titleStyle: {
        size: '16px', color: 'grey',
        fontFamily: 'Segoe UI', fontWeight: 'bold'
      },
      labelStyle: {
        size: '14px', color: 'blue',
```



```

        fontFamily : 'Segoe UI', fontWeight : 'bold'
    }
};
    this.title = 'Olympic Medals';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customizing Specific Point

You can customize the specific text in the axis labels using `axisLabelRender` event.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { categoryData } from './datasource';
import { IAxisLabelRenderEventArgs } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
(axisLabelRender) = 'axisLabelRender($event)'
[legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public legendSettings: Object = { visible: false };
  primaryYAxis: any;
  public axisLabelRender(args : IAxisLabelRenderEventArgs ): void {
    if(args.text === 'France') {

```

```

        args.labelStyle.color = 'Red';
    }
};
ngOnInit(): void {
    this.chartData = categoryData;
    this.primaryXAxis = {
        valueType: 'Category',
        title: 'Countries'
    };
    this.title = 'Olympic Medals';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Line break support

Line break feature used to customize the long axis label text into multiple lines by using tag. Refer the below example in that dataSource x value contains long text, it breaks into two lines by using `
` tag.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { Browser } from '@syncfusion/ej2-base';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='data' type='Bar' xName='x' yName='y'
width=2 ></e-series>
    </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public primaryYAxis?: Object;
    public chartData?: Object[];
}

```

```

public title?: string;
public data: Object[] = [
  { x: 'Germany', y: 72, country: 'GER: 72' },
  { x: 'Russia', y: 103.1, country: 'RUS: 103.1' },
  { x: 'Brazil', y: 139.1, country: 'BRZ: 139.1' },
  { x: 'India', y: 462.1, country: 'IND: 462.1' },
  { x: 'China', y: 721.4, country: 'CHN: 721.4' },
  { x: 'United States<br>Of America', y: 286.9, country: 'USA: 286.9' }
],
  { x: 'Great Britain', y: 115.1, country: 'GBR: 115.1' },
  { x: 'Nigeria', y: 97.2, country: 'NGR: 97.2' }
];
ngOnInit(): void {
  this.primaryXAxis = {
    title: 'Country',
    valueType: 'Category',
    majorGridLines: { width: 0 },
    enableTrim: false,
  };
  this.primaryYAxis = {
    minimum: 0,
    maximum: 800,
    labelFormat: Browser.isDevice ? '{value}' : '{value}M',
    edgeLabelPlacement: 'Shift',
    majorGridLines: { width: 0 },
    majorTickLines: { width: 0 },
    lineStyle: { width: 0 },
    labelStyle: {
      color: 'transparent'
    }
  };
  this.title = 'Internet Users - 2016';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Maximum Labels

[Link to the Video](#) property is set, then the labels will be rendered based on the count in the property per 100 pixel. If you have set range (minimum, maximum, interval) and maximumLabels, then the priority goes to range only. If you haven't set the range, then we have considered priority to maximumLabels property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'

```

```

import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { series1 } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart style='display:block' align='center'
id='chartcontainer' [title]='title' [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis'
[tooltip]='tooltip' [chartArea]='chartArea' [width]='width'>
    <e-series-collection>
      <e-series [dataSource]='series1' type='Line' xName='x'
yName='y' name='Germany' width=2 > </e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public series1: Object[] = series1;
  //Initializing Primary X Axis
  public primaryXAxis: Object = {
    title: 'Years',
    edgeLabelPlacement: 'Shift',
    majorGridLines : { width : 0 },
    maximumLabels:1,
  };
  //Initializing Primary Y Axis
  public primaryYAxis: Object = {
    title: 'Profit ($)',
    rangePadding: 'None',
    lineStyle : { width: 0 },
    majorTickLines : {width : 0}
  };
  public chartArea: Object = {
    border: {
      width: 0
    }
  };
  public width: string = '60%';
  public tooltip: Object = {
    enable: true
  };
  // custom code end
  public title: string = 'Inflation - Consumer Price';
  constructor() {
    //code
  }
  ngOnInit(): void {
    throw new Error('Method not implemented.');
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Axis customization in Angular Chart component

To know about axis customization, you can check on this video:

Axis Crossing

An axis can be positioned in the chart area using `crossesAt` and `crossesInAxis` properties. The `crossesAt` property specifies the values (datetime, numeric, or logarithmic) at which the axis line has to be intersected with the vertical axis or vice-versa, and the `crossesInAxis` property specifies the axis name with which the axis line has to be crossed.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { categoryData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = categoryData
    this.primaryXAxis = {
      valueType: 'Category',
      crossesAt : 15
    };
  };
}
```

```

        this.primaryYAxis = {
            crossesAt : 5
        };
        this.title = 'Olympic Medals';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Title

You can add a title to the axis using [title](#) property to provide quick information to the user about the data plotted in the axis.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChartModule } from '@syncfusion/ej2-angular-charts';
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
import { categoryData } from './datasource';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
    </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    ngOnInit(): void {
        this.chartData = categoryData;
        this.primaryXAxis = {
            valueType: 'Category',
            title: 'Countries'

```

```

    };
    this.primaryYAxis = {
      minimum: 0, maximum: 80,
      interval: 20, title: 'Medals',
      labelFormat: '${value}K',
      titleStyle: {
        size: '16px', color: 'grey',
        fontFamily: 'Segoe UI', fontWeight: 'bold'
      }
    };
    this.title = 'Olympic Medals';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Title Rotation

By using the [titleRotation](#) property, you can rotate the axis title from 0 to 360 degree.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChartModule } from '@syncfusion/ej2-angular-charts';
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
import { categoryData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;

```

```

ngOnInit(): void {
  this.chartData = categoryData;
  this.primaryXAxis = {
    valueType: 'Category',
    title: 'Countries'
  };
  this.primaryYAxis = {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals',
    labelFormat: '${value}K',
    titleStyle: {
      size: '16px', color: 'grey',
      fontFamily: 'Segoe UI', fontWeight: 'bold'
    }
  };
  this.title = 'Olympic Medals';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Tick Lines Customization

You can customize the **width**, **color** and **size** of the minor and major tick lines, using [majorTickLines](#) and [minorTickLines](#) properties in the axis.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChartModule } from '@syncfusion/ej2-angular-charts';
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
import { tickData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Sales'></e-series>
  </e-series-collection>
`
})
export class AppComponent implements OnInit {
  primaryXAxis: any;
  primaryYAxis: any;
  title: string;

  ngOnInit(): void {
    this.primaryXAxis = {
      valueType: 'Category',
      title: 'Countries'
    };
    this.primaryYAxis = {
      minimum: 0, maximum: 80,
      interval: 20, title: 'Medals',
      labelFormat: '${value}K',
      titleStyle: {
        size: '16px', color: 'grey',
        fontFamily: 'Segoe UI', fontWeight: 'bold'
      }
    };
    this.title = 'Olympic Medals';
  }
}

```



```

    </ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    ngOnInit(): void {
      this.chartData = tickData;
      this.primaryXAxis = {
        valueType: 'Category',
        majorTickLines : {
          color : 'blue',
          width : 5
        },
        minorTickLines : {
          color : 'red',
          width : 0
        }
      };
      this.primaryYAxis = {
        title: 'Temperature (Fahrenheit)',
        majorTickLines : {
          color : 'blue',
          width : 5
        },
        minorTickLines : {
          color : 'red',
          width : 0
        }
      };
      this.title = 'Temperature flow over months';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Grid Lines Customization

You can customize the **width**, **color** and **dashArray** of the minor and major grid lines, using [majorGridLines](#) and [minorGridLines](#) properties in the axis.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';

```

```

import { tickData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
    LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
    SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Sales'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = tickData;
    this.primaryXAxis = {
      valueType: 'Category',
      majorGridLines : {
        color : 'blue',
        width : 1
      },
      minorGridLines : {
        color : 'red',
        width : 0
      }
    };
    this.primaryYAxis = {
      title: 'Temperature (Fahrenheit)',
      majorGridLines : {
        color : 'blue',
        width : 1
      },
      minorGridLines : {
        color : 'red',
        width : 0
      }
    };
    this.title = 'Temperature flow over months';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';

```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Multiple Axis

In addition to primary X and Y axis, we can add n number of axis to the chart. Series can be associated with this axis, by mapping with axis's unique name.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { multipleData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-axes>
    <e-axis rowIndex=0 name='yAxis1' opposedPosition=true'
title='Temperature (Celsius)' [majorGridLines]='majorGridLines'
labelFormat='{value}°C'
[minimum]='24' [maximum]='36' [interval]='2'
[lineStyle]='lineStyle'>
    </e-axis>
  </e-axes>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Germany'></e-series>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y1' yAxisName='yAxis1' name='Japan' [marker]='marker'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  primaryYAxis: any;
  lineStyle: any;
  majorGridLines: any;
  ngOnInit(): void {
    this.chartData = multipleData;
    this.primaryXAxis = {
      valueType: 'Category'
    };
  }
}
```

```

        this.marker = {
            visible: true, width: 10, height: 10, border: { width: 2, color:
            '#F8AB1D' }
        }
        this.title = 'Weather Condition';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Inversed Axis

<!-- markdownlint-disable MD033 -->

When an axis is inversed, highest value of the axis comes closer to origin and vice versa. To place an axis in inversed manner set this property [isInversed](#) to true.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { inverseData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [title]='title'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[legendSettings]='legend'>
    <e-series-collection>
      <e-series [dataSource]='data' type='Column' xName='x'
yName='y' name='Years' [marker]='marker'>
    </e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent {
  public primaryXAxis?: Object;
  public primaryYAxis?: Object;
  public data?: Object[];
  public title?: string
  public legend: any;
}

```

```

public marker: any;
ngOnInit(): void {
  this.primaryYAxis = {
    isInversed: true
  };
  this.data= inverseData;
  this.title= 'Exchange Rate';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Opposed Position

<!-- markdownlint-disable MD012 -->

To place an axis opposite from its original position, set [opposedPosition](#) property of the axis to true.

<!-- markdownlint-disable MD012 -->

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { tickData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Sales'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
}

```

```

ngOnInit(): void {
  this.chartData = tickData;
  this.primaryXAxis = {
    valueType: 'Category'
  };
  this.primaryYAxis = {
    title: 'Temperature (Fahrenheit)',
    opposedPosition: true
  };
  this.title = 'Temperature flow over months';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

Strip line in Angular Chart component

<!-- markdownlint-disable MD036 -->

EJ2 chart supports horizontal and vertical strip lines and customization of stripline in both orientation.

To use Stripline in axis, we need to inject `StriplineService` into the `@NgModule.providers`

Horizontal Strip lines

You can create Horizontal stripline by adding the `stripline` in the vertical axis and set `visible` option to true.

Striplines are rendered in the specified start to end range and you can add more than one stripline for an axis.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { StripLineService, ColumnSeriesService, DataLabelService,
LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stripData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ StripLineService, ColumnSeriesService, DataLabelService,
LineSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
<e-series-collection>

```

```

        <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Internet'></e-series>>
    </e-series-collection>
</ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public primaryYAxis?: Object;
        ngOnInit(): void {
            this.chartData = [{x: 1, y: 20},{x: 2, y: 22},{x: 3, y: 0},{x: 4,
y: 12},{x: 5, y: 5},
                {x: 6, y: 15},{x: 7, y: 6},{x: 8, y: 12},{x: 9, y: 20},{x:
10, y: 7}];
            this.primaryYAxis = {
                title: 'Runs',
                stripLines:[
                    { start: 15, end: 22, text: 'Good', color: '#ff512f', visible:
true, zIndex: 'Behind', opacity: 0.5 },
                    { start: 8, end: 15, text: 'Medium', color: 'pink', opacity:
0.5, visible: true, zIndex: 'Behind' },
                    { start: 0, end: 8, text: 'Not enough', color: 'skyblue',
opacity: 0.5, visible: true, zIndex: 'Behind' }]
            };
            this.primaryXAxis = {
                title: 'Overs'
            };
            this.title = 'India Vs Australia 1st match';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Vertical Striplines

You can create vertical stripline by adding `thestripline` in the horizontal axis and set `visible` option to `true`.

Striplines are rendered in the specified start to end range and you can add more than one stripline for an axis.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { StripLineService, ColumnSeriesService, DataLabelService,
LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stripData } from './datasource';

```

```

@Component({
  imports: [
    ChartModule
  ],
  providers: [ StripLineService, ColumnSeriesService, DataLabelService,
    LineSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
      <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Internet'></e-series>>
      </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = [{x: 1, y: 20},{x: 2, y: 22},{x: 3, y: 0},{x: 4,
y: 12},{x: 5, y: 5},
                        {x: 6, y: 15},{x: 7, y: 6},{x: 8, y: 12},{x: 9, y: 20},{x:
10, y: 7}];
    this.primaryYAxis = {
      title: 'Runs',
    };
    this.primaryXAxis = {
      title: 'Overs',
      stripLines:[
        {start: 0, end: 5, text: 'powerplay 1', color: 'red', visible:
true, opacity: 0.5, rotation: 45, textStyle: { size: 20, color: 'black'}},
        {start: 5, end: 10, text: 'powerplay 2', color: 'blue', visible:
true, opacity: 0.5, rotation: 45, textStyle: { size: 20, color: 'black'}},
      ]
    };
    this.title = 'India Vs Australia 1st match';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customize the strip line

Starting value in specific strip line can be customized by `start` property in strip line. Similarly, ending value is customized by `end`. It can be also set for starting from the corresponding origin of the axis by `startFromOrigin`.

Size of the strip line is customized by `size`. Border for the stripline is customized by `border`. Order of the strip line such that whether it should be rendered in behind or over the series elements is customized by `zIndex`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { StripLineService, ColumnSeriesService, DataLabelService,
LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stripData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ StripLineService, ColumnSeriesService, DataLabelService,
LineSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Internet'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = [
      {x: 1, y: 20}, {x: 2, y: 22}, {x: 3, y: 0}, {x: 4, y: 12}, {x:
5, y: 5},
      {x: 6, y: 15}, {x: 7, y: 6}, {x: 8, y: 12}, {x: 9, y: 20}, {x:
10, y: 7}
    ];
    this.primaryYAxis = {
      title: 'Runs',
    };
    this.primaryXAxis = {
      stripLines:[
        { startFromOrigin: true, size: 4, zIndex: 'Behind', opacity:
0.5, border: { width: 2, color:'red'}}
      ],
      title: 'Overs'
    };
    this.title = 'India Vs Australia 1st match';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customize the stripline text

You can customize the text rendered in stripline by `textStyle` property. Rotation of the strip line text can be changed by `rotation` property.

Horizontal and Vertical alignment of stripline text can be changed by `horizontalAlignment` and `verticalAlignment` property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { StripLineService, ColumnSeriesService, DataLabelService,
LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stripData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ StripLineService, ColumnSeriesService, DataLabelService,
LineSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Internet'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = [
      {x: 1, y: 20}, {x: 2, y: 22}, {x: 3, y: 0}, {x: 4, y: 12}, {x:
5, y: 5},
      {x: 6, y: 15}, {x: 7, y: 6}, {x: 8, y: 12}, {x: 9, y: 20}, {x:
10, y: 7}
    ];
    this.primaryYAxis = {
      title: 'Runs',
    };
    this.primaryXAxis = {
      stripLines:[
```

```

        { startFromOrigin: true, size: 4, zIndex: 'Behind', opacity:
0.5, text: 'Good', verticalAlignment: 'Middle', horizontalAlignment:
'Middle', rotation: 90, textStyle: { size: 15}},
        { start: 5, end: 8, verticalAlignment: 'Start',
horizontalAlignment: 'End', rotation: 45, text: 'Poor'}
    ],
    title: 'Overs'
  };
  this.title = 'India Vs Australia 1st match';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Dash Array

You can create dash array stripline by using `dashArray` property. The Striplines are rendered with specified dash array values.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { StripLineService, ColumnSeriesService, DataLabelService,
LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stripData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ StripLineService, ColumnSeriesService, DataLabelService,
LineSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryYAxis]='primaryYAxis'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' [marker]='marker'></e-series>>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public primaryYAxis?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.chartData = [

```

```

        {x: 1, y: 5}, {x: 2, y: 39}, {x: 3, y: 21}, {x: 4, y: 51}, {x: 5, y:
30},
        {x: 6, y: 25}, {x: 7, y: 10}, {x: 8, y: 40}, {x: 9, y: 50}, {x: 10,
y: 20}
    ];
    this.primaryYAxis = {
        minimum: 0, maximum: 60, interval: 10,
        stripLines: [
            { start: 30, size: 2, sizeType: 'Pixel', dashArray: "10,5",
color: "red" }
        ]
    };
    this.marker = { visible: true }
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Recurrence Stripline

The strip lines to be drawn repeatedly at the regular intervals – this will be useful when you want to mark an event that occurs recursively along the timeline of the chart. Following properties are used to configure this feature.

- **isRepeat** - It is used for enable / disable the recurrence strip line.
- **repeatEvery** - It is used for mention the stripline interval.
- **repeatUntil** - It specifies the end value at which point strip line has to stop repeating.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { StripLineService, ColumnSeriesService, DataLabelService,
LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stripData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ StripLineService, ColumnSeriesService, DataLabelService,
LineSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'>
    <e-series-collection>

```

```

        <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' [marker]='marker'></e-series>>
    </e-series-collection>
</ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public marker?: Object;
        ngOnInit(): void {
            this.chartData = [
                {x: 1, y: 5},{x: 2, y: 39},{x: 3, y: 21},{x: 4, y: 51},{x: 5, y:
30},
                {x: 6, y: 25},{x: 7, y: 10},{x: 8, y: 40},{x: 9, y: 50},{x: 10,
y: 20}
            ];
            this.primaryXAxis = {
                stripLines:[
                    {start: 1, size: 1, isRepeat: true, repeatEvery: 2, color:
'rgba(167,169,171, 0.3)'}
                ]
            };
            this.marker = { visible: true }
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Size Type

The `sizeType` property refers the size of the stripline. They are,

- Auto
- Pixel
- Years
- Months
- Days
- Hours
- Minutes
- Seconds

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService } from '@syncfusion/ej2-angular-charts'

```

```

import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
    StripLineService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
    [primaryYAxis]='primaryYAxis'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='x'
        yName='y' [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public primaryYAxis?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.chartData = [{ x: new Date(2000, 0, 1), y: 10 }, { x: new
    Date(2002, 0, 1), y: 40 },
    { x: new Date(2004, 0, 1), y: 20 }, { x: new Date(2006, 0, 1), y: 50 },
    { x: new Date(2008, 0, 1), y: 15 }, { x: new Date(2010, 0, 1), y: 30 }];
    this.primaryXAxis = {
      valueType: 'DateTime', intervalType: 'Years',
      stripLines:[
        {start:new Date(2002, 0, 1) , size: 2, sizeType: 'Years', color:
        'rgba(167,169,171, 0.3)'}
      ]
    };
    this.primaryYAxis = {
      minimum: 0, maximum: 60, interval: 10
    };
    this.marker = { visible: true }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Segment Stripline

You can create stripline in a particular region with respect to segment. You can enable the segment stripline using `isSegmented` property. The start and end value of this type of stripline can be defined using `segmentStart` and `segmentEnd` properties.

- `isSegmented` - It is used to enable the segment stripline.

- **segmentStart** - Used to change the segment start value. Value correspond to associated axis.
- **segmentEnd** - Used to change the segment end value. Value correspond to associated axis.
- **segmentAxisName** - Used to specify the name of the associated axis.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { StripLineService, ColumnSeriesService, DataLabelService,
LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stripData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ StripLineService, ColumnSeriesService, DataLabelService,
LineSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryYAxis]='primaryYAxis'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public primaryYAxis?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.chartData = [
      {x: 1, y: 5},{x: 2, y: 39},{x: 3, y: 21},{x: 4, y: 51},{x: 5, y:
30},
      {x: 6, y: 25},{x: 7, y: 10},{x: 8, y: 40},{x: 9, y: 50},{x: 10,
y: 20}
    ];
    this.primaryYAxis = {
      stripLines:[
        {start: 20, end: 40, isSegmented: true, segmentStart: 2,
segmentEnd: 4,
        color: 'rgba(167,169,171, 0.3)'}
      ]
    };
    this.marker = { visible: true }
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
```

```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Mark the threshold in chart](#)

Multiple panes in Angular Chart component

Chart area can be divided into multiple panes using [rows](#) and [columns](#).

Rows

To split the chart area vertically into number of rows, use [rows](#) property of the chart.

- You can allocate space for each row by using the [height](#) property. The value can be either in percentage or in pixel.
- To associate a vertical axis to a particular row, specify its index to [rowIndex](#) property of the axis.
- To customize each row's bottom line, use [border](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService, LineSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, ColumnSeriesService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-axes>
    <e-axis rowIndex=1 name='yAxis1' opposedPosition=true'
title='Temperature (Celsius)' [majorGridLines]='majorGridLines'
labelFormat='{value}°C'
[minimum]='24' [maximum]='36' [interval]='2'
[lineStyle]='lineStyle'>
  </e-axis>
</e-axes>
<e-rows>
  <e-row height=50%></e-row>
  <e-row height=50%></e-row>
</e-rows>
<e-series-collection>
  <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Germany'></e-series>
  <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y1' name='Japan' yAxisName='yAxis1' [marker]='marker'></e-series>
```



```

        </e-series-collection>
    </ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public majorGridLines?: Object;
        public primaryYAxis?: Object;
        public lineStyle?: Object;
        public marker?: Object;
        public rows?: Object;
        ngOnInit(): void {
            this.chartData = [
                { x: 'Jan', y: 15, y1: 33 }, { x: 'Feb', y: 20, y1: 31 }, {
x: 'Mar', y: 35, y1: 30 },
                { x: 'Apr', y: 40, y1: 28 }, { x: 'May', y: 80, y1: 29 }, {
x: 'Jun', y: 70, y1: 30 },
                { x: 'Jul', y: 65, y1: 33 }, { x: 'Aug', y: 55, y1: 32 }, {
x: 'Sep', y: 50, y1: 34 },
                { x: 'Oct', y: 30, y1: 32 }, { x: 'Nov', y: 35, y1: 32 }, {
x: 'Dec', y: 35, y1: 31 }
            ];
            this.primaryXAxis = {
                title: 'Months',
                valueType: 'Category',
                interval: 1
            };
            this.primaryYAxis = {
                minimum: 0, maximum: 90, interval: 20,
                lineStyle: { width: 0 },
                title: 'Temperature (Fahrenheit)',
                labelFormat: '{value}°F'
            };
            this.majorGridLines = { width: 0 };
            this.lineStyle = { width: 0 };
            this.marker = {
                visible: true, width: 10, height: 10, border: { width: 2, color:
'#F8AB1D' }
            };
            this.title = 'Weather Condition';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

For spanning the vertical axis along multiple row, you can use [span](#) property of an axis.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'

```

```

import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, ColumnSeriesService, LineSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-axes>
      <e-axis rowIndex=1 name='yAxis1' opposedPosition=true'
title='Temperature (Celsius)' [majorGridLines]='majorGridLines'
labelFormat='{value}°C'
[minimum]='24' [maximum]='36' [interval]='2'
[lineStyle]='lineStyle'>
    </e-axis>
  </e-axes>
  <e-rows>
    <e-row height=50%></e-row>
    <e-row height=50%></e-row>
  </e-rows>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Germany'></e-series>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y1' name='Japan' yAxisName='yAxis1' [marker]='marker'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public majorGridLines?: Object;
  public primaryYAxis?: Object;
  public lineStyle?: Object;
  public marker?: Object;
  public rows?: Object;
  ngOnInit(): void {
    this.chartData = [
      { x: 'Jan', y: 15, y1: 33 }, { x: 'Feb', y: 20, y1: 31 }, {
x: 'Mar', y: 35, y1: 30 },
      { x: 'Apr', y: 40, y1: 28 }, { x: 'May', y: 80, y1: 29 }, {
x: 'Jun', y: 70, y1: 30 },
      { x: 'Jul', y: 65, y1: 33 }, { x: 'Aug', y: 55, y1: 32 }, {
x: 'Sep', y: 50, y1: 34 },
      { x: 'Oct', y: 30, y1: 32 }, { x: 'Nov', y: 35, y1: 32 }, {
x: 'Dec', y: 35, y1: 31 }
    ];
    this.primaryXAxis = {
      title: 'Months',
      valueType: 'Category',

```

```

        interval: 1
    };
    this.primaryYAxis = {
        minimum: 0, maximum: 90, interval: 20,
        lineStyle: { width: 0 },
        title: 'Temperature (Fahrenheit)',
        labelFormat: '{value}°F',
        span: 2
    };
    this.majorGridLines = { width: 0 };
    this.lineStyle = { width: 0 };
    this.marker = {
        visible: true, width: 10, height: 10, border: { width: 2, color:
'#F8AB1D' }
    }
    this.title = 'Weather Condition';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Columns

To split the chart area horizontally into number of columns, use [columns](#) property of the chart.

- You can allocate space for each column by using the [width](#)

property. The given width can be either in percentage or in pixel.

- To associate a horizontal axis to a particular column, specify its index to [columnIndex](#) property of the axis.
- To customize each column's bottom line, use [border](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService, LineSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, ColumnSeriesService, LineSeriesService ],
  standalone: true,
  selector: 'app-container',

```

```

    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-axes>
        <e-axis columnIndex=1 name='xAxis1' opposedPosition='true'
[majorGridLines]='majorGridLines'
            valueType='Category' [lineStyle]='lineStyle'>
        </e-axis>
    </e-axes>
    <e-columns>
        <e-column width=50%></e-column>
        <e-column width=50%></e-column>
    </e-columns>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Germany'></e-series>
        <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y1' name='Japan' xAxisName='xAxis1' [marker]='marker'></e-series>
    </e-series-collection>
    </ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public majorGridLines?: Object;
    public primaryYAxis?: Object;
    public lineStyle?: Object;
    public marker?: Object;
    public rows?: Object;
    ngOnInit(): void {
      this.chartData = [
        { x: 'Jan', y: 15, y1: 33 }, { x: 'Feb', y: 20, y1: 31 }, {
x: 'Mar', y: 35, y1: 30 },
        { x: 'Apr', y: 40, y1: 28 }, { x: 'May', y: 80, y1: 29 }, {
x: 'Jun', y: 70, y1: 30 },
        { x: 'Jul', y: 65, y1: 33 }, { x: 'Aug', y: 55, y1: 32 }, {
x: 'Sep', y: 50, y1: 34 },
        { x: 'Oct', y: 30, y1: 32 }, { x: 'Nov', y: 35, y1: 32 }, {
x: 'Dec', y: 35, y1: 31 }
      ];
      this.primaryXAxis = {
        title: 'Months',
        valueType: 'Category',
        interval: 1
      };
      this.primaryYAxis = {
        minimum: 0, maximum: 90, interval: 20,
        lineStyle: { width: 0 },
        title: 'Temperature (Fahrenheit)',
        labelFormat: '{value}°F'
      };
      this.majorGridLines = { width: 0 };
      this.lineStyle = { width: 0 };
      this.marker = {
        visible: true, width: 10, height: 10, border: { width: 2, color:
'#F8AB1D' }
      }
    }
  }

```

```

        this.title = 'Weather Condition';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

For spanning the horizontal axis along multiple column, you can use [span](#) property of an axis.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, ColumnSeriesService, LineSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-axes>
      <e-axis columnIndex=1 name='xAxis1' opposedPosition='true'
[majorGridLines]='majorGridLines'
        valueType='Category' [lineStyle]='lineStyle'>
      </e-axis>
    </e-axes>
    <e-columns>
      <e-column width=50%></e-column>
      <e-column width=50%></e-column>
    </e-columns>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Germany'></e-series>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y1' xAxisName='xAxis1' name='Japan' [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public majorGridLines?: Object;
  public primaryYAxis?: Object;
  public lineStyle?: Object;
  public marker?: Object;
}

```

```

public rows?: Object;
ngOnInit(): void {
    this.chartData = [
        { x: 'Jan', y: 15, y1: 33 }, { x: 'Feb', y: 20, y1: 31 }, {
x: 'Mar', y: 35, y1: 30 },
        { x: 'Apr', y: 40, y1: 28 }, { x: 'May', y: 80, y1: 29 }, {
x: 'Jun', y: 70, y1: 30 },
        { x: 'Jul', y: 65, y1: 33 }, { x: 'Aug', y: 55, y1: 32 }, {
x: 'Sep', y: 50, y1: 34 },
        { x: 'Oct', y: 30, y1: 32 }, { x: 'Nov', y: 35, y1: 32 }, {
x: 'Dec', y: 35, y1: 31 }
    ];
    this.primaryXAxis = {
        title: 'Months',
        valueType: 'Category',
        interval: 1,
        span: 2
    };
    this.primaryYAxis = {
        minimum: 0, maximum: 90, interval: 20,
        lineStyle: { width: 0 },
        title: 'Temperature (Fahrenheit)',
        labelFormat: '{value}°F'
    };
    this.majorGridLines = { width: 0 };
    this.lineStyle = { width: 0 };
    this.marker = {
        visible: true, width: 10, height: 10, border: { width: 2, color:
'#F8AB1D' }
    };
    this.title = 'Weather Condition';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Chart Types

Line Chart in Angular Chart component

Line

To render a line series, use series [type](#) as `Line` and inject `LineSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, StepLineSeriesService,
SplineSeriesService, StackingLineSeriesService, DateTimeService,

```

```

    SplineAreaSeriesService, MultiColoredLineSeriesService,
    ParetoSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-
    charts'
import { Component, OnInit } from '@angular/core';
import { lineData } from '../datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService, StepLineSeriesService,
    SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
    ParetoSeriesService, ColumnSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis'
  [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public title?: string;
  public primaryXAxis?: Object;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = lineData;
    this.primaryXAxis = {
      interval: 1
    };
    this.primaryYAxis =
    {
      title: 'Expense',
    },
    this.title = 'Efficiency of oil-fired power production';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Multicolored Line

To render a multicolored line series, use the series type as `MultiColoredLine`, and inject the `MultiColoredLineSeriesService` into the `@NgModule.providers`.

Here, the individual colors to the data can be mapped by using `pointColorMapping`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, StepLineSeriesService,
SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
ParetoSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { splineData } from './datasource';
@Component({
imports: [
    ChartModule
],
providers: [ CategoryService, LineSeriesService, StepLineSeriesService,
SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
ParetoSeriesService, ColumnSeriesService],
standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='MultiColoredLine'
xName='x' yName='y' name='London' width=2 [marker]='marker'
            pointColorMapping= 'color'></e-series>
    </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
    public chartData?: Object[];
    public title?: string;
    public marker?: Object;
    primaryXAxis: any;
    primaryYAxis: any;
    ngOnInit(): void {
        this.chartData = [{ x: 2005, y: 28 , color: 'red'}, { x: 2006, y:
25, color:'green'},
            { x: 2007, y: 26, color: '#ff0097' }, { x: 2008, y: 27, color:
'crimson' },
            { x: 2009, y: 32, color: 'blue' }, { x: 2010, y: 35 ,color:
'darkorange'}];
        this.marker = { visible: true, width: 10, height: 10 };
        this.title = 'Climate Graph-2012';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```


Series customization

The following properties can be used to customize the **line** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes for series.
- [width](#) – Specifies the width for series.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, StepLineSeriesService,
SplineSeriesService, StackingLineSeriesService, DateTimeService,
SplineAreaSeriesService, MultiColoredLineSeriesService,
ParetoSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { lineData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService, StepLineSeriesService,
SplineSeriesService, StackingLineSeriesService, DateTimeService,
SplineAreaSeriesService, MultiColoredLineSeriesService,
ParetoSeriesService, ColumnSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='India' fill='green' width=3 dashArray='5,5'
[marker]='marker'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = lineData;
    this.marker = { visible: true };
    this.title = 'Efficiency of oil-fired power production';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
```

```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Data label](#)
- [Tooltip](#)

Step line chart in Angular Chart component

Step line

To render a step line series, use series [type](#) as `StepLine` and inject `StepLineSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, StepLineSeriesService,
  SplineSeriesService, StackingLineSeriesService, DateTimeService,
  SplineAreaSeriesService, MultiColoredLineSeriesService,
  ParetoSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { data } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService, StepLineSeriesService,
    SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
    ParetoSeriesService, ColumnSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y' name='USA' width=2 [marker]='marker'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public title?: string;
  primaryXAxis: any;
  primaryYAxis: any;
  marker: any;
  ngOnInit(): void {
    this.chartData = data;
    this.title = 'CO2 - Intensity Analysis';
  }
}
```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Series customization

The following properties can be used to customize the **step line** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes for series.
- [width](#) – Specifies the width for series.
- [step](#) – Specifies the position of the step for the series.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, StepLineSeriesService,
    SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
    ParetoSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { data } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService, StepLineSeriesService,
    SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
    ParetoSeriesService, ColumnSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y' name='USA' width=2 [marker]='marker' fill='blue' opacity=0.6
step='Left' dashArray='5.0'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public title?: string;
  primaryXAxis: any;
  primaryYAxis: any;
```

```

marker: any;
ngOnInit(): void {
    this.chartData = data;
    this.title = 'CO2 - Intensity Analysis';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See also

- [Data label](#)
- [Tooltip](#)

Stacked Line Chart in Angular Chart component

Stacked Line

To render a stacked line series, use series [type](#) as `StackingLine` and inject `StackingLineSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, StepLineSeriesService,
    SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
    ParetoSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { data0, data1, data2, data3 } from './datasource';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ CategoryService, LineSeriesService, StepLineSeriesService,
        SplineSeriesService, StackingLineSeriesService, DateTimeService,
        SplineAreaSeriesService, MultiColoredLineSeriesService,
        ParetoSeriesService, ColumnSeriesService ],
    standalone: true,
    selector: 'app-container',
    template: `<ej-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'>
    <e-series-collection>
        <e-series [dataSource]='chartData1' type='StackingLine'
xName='x' yName='y' name='John' width='2' [marker]='marker' dashArray='5,1'>
    </e-series>

```

```

        <e-series [dataSource]='chartData2' type='StackingLine'
xName='x' yName='y' name='Peter' width='2' [marker]='marker'
dashArray='5,1'> </e-series>
        <e-series [dataSource]='chartData3' type='StackingLine'
xName='x' yName='y' name='Steve' width='2' [marker]='marker'
dashArray='5,1'> </e-series>
        <e-series [dataSource]='chartData4' type='StackingLine'
xName='x' yName='y' name='Charle' width='2' [marker]='marker'
dashArray='5,1'> </e-series>
    </e-series-collection>
</ejs-chart>`
}))
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public title?: string;
    public primaryYAxis?: Object;
    public marker?: Object;
    public series?: Object;
    public chartData1?: Object[]; public chartData2?: Object[]; public
chartData3?: Object[];
    public chartData4?: Object[];
    ngOnInit(): void {
        this.primaryXAxis = {
            interval: 1, valueType: 'Category'
        };
        this.primaryYAxis =
        {
            title: 'Expense',
            interval: 100,
            labelFormat: '${value}',
        },
        this.chartData1 = data0;
        this.chartData2 = data1;
        this.chartData3 = data2;
        this.chartData4 = data3;
        this.marker = { visible: true };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

The following properties can be used to customize the **stacked line** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes for series.
- [width](#) – Specifies the width for series.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, StepLineSeriesService,
SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
ParetoSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { data0, data1, data2, data3 } from './datasource';
@Component({
imports: [
    ChartModule
],
providers: [ CategoryService, LineSeriesService, StepLineSeriesService,
SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
ParetoSeriesService, ColumnSeriesService],
standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'>
    <e-series-collection>
        <e-series [dataSource]='chartData1' type='StackingLine'
xName='x' yName='y' name='John' width='2' [marker]='marker' dashArray='5'
fill='red'> </e-series>
        <e-series [dataSource]='chartData2' type='StackingLine'
xName='x' yName='y' name='Peter' width='2' [marker]='marker' dashArray='5'
fill='yellow'> </e-series>
        <e-series [dataSource]='chartData3' type='StackingLine'
xName='x' yName='y' name='Steve' width='2' [marker]='marker' dashArray='5'
fill='black'> </e-series>
        <e-series [dataSource]='chartData4' type='StackingLine'
xName='x' yName='y' name='Charle' width='2' [marker]='marker' dashArray='5'
fill='blue'> </e-series>
    </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public title?: string;
    public primaryYAxis?: Object;
    public marker?: Object;
    public series?: Object;
    public chartData1?: Object[]; public chartData2?: Object[]; public
chartData3?: Object[];
    public chartData4?: Object[];
    ngOnInit(): void {
        this.primaryXAxis = {
            interval: 1, valueType: 'Category'
        };
        this.primaryYAxis =
        {
            title: 'Expense',
            interval: 100,

```

```

        labelFormat: '${value}',
    },
    this.chartData1 = data0;
    this.chartData2 = data1;
    this.chartData3 = data2;
    this.chartData4 = data3;
    this.marker = { visible: true };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Data label](#)
- [Tooltip](#)

100% Stacked Line Chart in Angular Chart component

100% Stacked Line

To render a 100% stacked line series, use series [type](#) as `StackingLine100` and inject `StackingLineSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, StepLineSeriesService,
SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
ParetoSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { data0, data1, data2, data3 } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService, StepLineSeriesService,
SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
ParetoSeriesService, ColumnSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'>
    <e-series-collection>

```

```

        <e-series [dataSource]='chartData1' type='StackingLine100'
xName='x' yName='y' name='John' width='2' [marker]='marker' dashArray='5,1'>
</e-series>
        <e-series [dataSource]='chartData2' type='StackingLine100'
xName='x' yName='y' name='Peter' width='2' [marker]='marker'
dashArray='5,1'> </e-series>
        <e-series [dataSource]='chartData3' type='StackingLine100'
xName='x' yName='y' name='Steve' width='2' [marker]='marker'
dashArray='5,1'> </e-series>
        <e-series [dataSource]='chartData4' type='StackingLine100'
xName='x' yName='y' name='Charle' width='2' [marker]='marker'
dashArray='5,1'> </e-series>
    </e-series-collection>
</ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public title?: string;
        public primaryYAxis?: Object;
        public marker?: Object;
        public series?: Object;
        public chartData1?: Object[]; public chartData2?: Object[]; public
chartData3?: Object[];
        public chartData4?: Object[];
        ngOnInit(): void {
            this.primaryXAxis = {
                interval: 1, valueType: 'Category'
            };
            this.primaryYAxis =
            {
                title: 'Expense',
                interval: 100,
                labelFormat: '${value}',
            },
            this.chartData1 = data0;
            this.chartData2 = data1;
            this.chartData3 = data2;
            this.chartData4 = data3;
            this.marker = { visible: true };
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

The following properties can be used to customize the 100% stacked line series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).

- [dashArray](#) – Specifies the dashes for series.
- [width](#) – Specifies the width for series.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, StepLineSeriesService,
    SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
    ParetoSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { data0, data1, data2, data3 } from './datasource';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ CategoryService, LineSeriesService, StepLineSeriesService,
        SplineSeriesService, StackingLineSeriesService, DateTimeService,
        SplineAreaSeriesService, MultiColoredLineSeriesService,
        ParetoSeriesService, ColumnSeriesService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'>
    <e-series-collection>
        <e-series [dataSource]='chartData1' type='StackingLine100'
xName='x' yName='y' name='John' width='2' [marker]='marker' dashArray='5,1'
fill='red'> </e-series>
        <e-series [dataSource]='chartData2' type='StackingLine100'
xName='x' yName='y' name='Peter' width='2' [marker]='marker' dashArray='5,1'
fill='yellow'> </e-series>
        <e-series [dataSource]='chartData3' type='StackingLine100'
xName='x' yName='y' name='Steve' width='2' [marker]='marker' dashArray='5,1'
fill='blue'> </e-series>
        <e-series [dataSource]='chartData4' type='StackingLine100'
xName='x' yName='y' name='Charle' width='2' [marker]='marker'
dashArray='5,1' fill='orange'> </e-series>
    </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public title?: string;
    public primaryYAxis?: Object;
    public marker?: Object;
    public series?: Object;
    public chartData1?: Object[]; public chartData2?: Object[]; public
chartData3?: Object[];
    public chartData4?: Object[];
    ngOnInit(): void {
        this.primaryXAxis = {
            interval: 1, valueType: 'Category'
        };
    }
}
```

```

        this.primaryYAxis =
        {
            title: 'Expense',
            interval: 100,
            labelFormat: '${value}',
        },
        this.chartData1 = data0;
        this.chartData2 = data1;
        this.chartData3 = data2;
        this.chartData4 = data3;
        this.marker = { visible: true };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Data label](#)
- [Tooltip](#)

Spline Chart in Angular Chart component

Spline

To render a spline series, use series [type](#) as **Spline** and inject **SplineSeriesService** into the **@NgModule.providers**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, StepLineSeriesService,
    SplineSeriesService, StackingLineSeriesService, DateTimeService,
        SplineAreaSeriesService, MultiColoredLineSeriesService,
    ParetoSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { splineData } from './datasource';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ CategoryService, LineSeriesService, StepLineSeriesService,
        SplineSeriesService, StackingLineSeriesService, DateTimeService,
            SplineAreaSeriesService, MultiColoredLineSeriesService,
        ParetoSeriesService, ColumnSeriesService],
    standalone: true,
    selector: 'app-container',

```

```

    template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
    [title]='title'>
      <e-series-collection>
        <e-series [dataSource]='chartData' type='Spline' xName='x'
        yName='y' name='London' width=2 [marker]='marker'></e-series>
      </e-series-collection>
    </ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    public marker?: Object;
    ngOnInit(): void {
      this.chartData = splineData;
      this.primaryXAxis = {
        title: 'Month',
        valueType: 'Category'
      };
      this.marker = { visible: true, width: 10, height: 10 };
      this.title = 'Climate Graph-2012';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

The following properties can be used to customize the **spline** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes for series.
- [width](#) – Specifies the width for series.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, StepLineSeriesService,
SplineSeriesService, StackingLineSeriesService, DateTimeService,
SplineAreaSeriesService, MultiColoredLineSeriesService,
ParetoSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { splineData } from './datasource';
@Component({
  imports: [

```

```

    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService, StepLineSeriesService,
    SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
    ParetoSeriesService, ColumnSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
    [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Spline' xName='x'
yName='y' name='London' width=2 [marker]='marker' fill='yellow'
dashArray='5.5'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.chartData = splineData;
    this.primaryXAxis = {
      title: 'Month',
      valueType: 'Category'
    };
    this.marker = { visible: true, width: 10, height: 10 };
    this.title = 'Climate Graph-2012';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Data label](#)
- [Tooltip](#)

Area Chart in Angular Chart component

Area

To render a area series, use series [type](#) as `Area` and inject `AreaSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'

```

```

import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeAreaSeriesService, StepAreaSeriesService,
StackingAreaSeriesService,
    DateTimeService, CategoryService, MultiColoredAreaSeriesService,
StackingStepAreaSeriesService, SplineRangeAreaSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { areaData } from './datasource';
@Component({
  imports: [
    ChartModule, ChartAllModule
  ],
  providers: [ AreaSeriesService , RangeAreaSeriesService,
    StepAreaSeriesService, StackingAreaSeriesService,
    DateTimeService, CategoryService,
    MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
    eriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Area' xName='x'
yName='y' name='Product A' fill='#69D2E7' opacity=0.6></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = areaData;
    this.primaryXAxis = {
      title: 'Year',
      minimum: 1900, maximum: 2000, interval: 10,
      edgeLabelPlacement: 'Shift'
    };
    this.primaryYAxis = {
      minimum: 2, maximum: 5, interval: 0.5,
      title: 'Sales Amount in Millions'
    };
    this.title = 'Average Sales Comparison';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Multicolored area

To render a multicolored area series, use the series type as `MultiColoredArea`, and inject the `MultiColoredAreaSeriesService` into the `@NgModule.providers`.

The required segments of the series can be customized using the `value`, `color`, and `dashArray`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeAreaSeriesService, StepAreaSeriesService,
  StackingAreaSeriesService,
  DateTimeService, CategoryService, MultiColoredAreaSeriesService,
  StackingStepAreaSeriesService, SplineRangeAreaSeriesService } from
  '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stepData } from '../datasource';
@Component({
  imports: [
    ChartModule, ChartAllModule
  ],
  providers: [ AreaSeriesService , RangeAreaSeriesService,
  StepAreaSeriesService, StackingAreaSeriesService,
    DateTimeService, CategoryService,
  MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
    eriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
  [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='MultiColoredArea'
  xName='x' yName='y' name='England'
      segmentAxis='X' [segments]='segments'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public segments?: Object[];
  ngOnInit(): void {
    this.chartData = [{ x: 2005, y: 28 }, { x: 2006, y: 25}, { x: 2007,
  y: 26 }, { x: 2008, y: 27 },
    { x: 2009, y: 32}, { x: 2010, y: 35 }, { x: 2011, y: 25 }];
    this.title = 'England - Run Rate';
    this.segments = [{
      value: 2007,
      color: 'blue'
    }, {
      value: 2009,
      color: 'lightgreen'
    }, {
      color: 'orange'
    }
  ]
}
```

```
    }];
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Series customization

The following properties can be used to customize the **area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeAreaSeriesService, StepAreaSeriesService,
  StackingAreaSeriesService,
  DateTimeService, CategoryService, MultiColoredAreaSeriesService,
  StackingStepAreaSeriesService, SplineRangeAreaSeriesService } from
  '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { areaData } from './datasource';
@Component({
  imports: [
    ChartModule, ChartAllModule
  ],
  providers: [ AreaSeriesService , RangeAreaSeriesService,
  StepAreaSeriesService, StackingAreaSeriesService,
    DateTimeService, CategoryService,
  MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
  eriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Area' xName='x'
yName='y' name='Product A' fill='green' width=2 dashArray='5,5'
[border]='border' opacity=0.6></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public title?: string;
  public border?: Object;
```

```

primaryXAxis: any;
primaryYAxis: any;
ngOnInit(): void {
    this.border = {
        color: 'red',
        width: 2
    };
    this.chartData = areaData;
    this.title = 'Average Sales Comparison';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Area border

The following properties in the [bordermodel](#) can be used to customize the border of the Area Chart.

[width](#) - Specifies the width for the border of the Area Chart.

[color](#) - Specifies the color for the border of the Area Chart.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeAreaSeriesService, StepAreaSeriesService,
StackingAreaSeriesService,
    DateTimeService, CategoryService, MultiColoredAreaSeriesService,
StackingStepAreaSeriesService, SplineRangeAreaSeriesService } from
'@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { areaData } from './datasource';
@Component({
  imports: [
    ChartModule, ChartAllModule
  ],
  providers: [ AreaSeriesService , RangeAreaSeriesService,
StackingAreaSeriesService, StackingAreaSeriesService,
    DateTimeService, CategoryService,
MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
eriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Area' xName='x'
yName='y' name='Product A' width=2 [border]='border' opacity=0.5></e-series>
    </e-series-collection>
</ejs-chart>`
})

```



```

}))
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public title?: string;
  public border?: Object;
  primaryXAxis: any;
  primaryYAxis: any;
  ngOnInit(): void {
    this.border = {
      width: 1.5
    };
    this.chartData = areaData;
    this.title = 'Average Sales Comparison';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Data label](#)
- [Tooltip](#)

Range Area in Angular Chart component

Range Area

To render a range area series, use series [type](#) as `RangeArea` and inject `RangeAreaSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeAreaSeriesService, StepAreaSeriesService,
  StackingAreaSeriesService,
  DateTimeService, CategoryService, MultiColoredAreaSeriesService,
  StackingStepAreaSeriesService, SplineRangeAreaSeriesService } from
  '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { rangeData } from './datasource';
@Component({
  imports: [
    ChartModule, ChartAllModule
  ],
  providers: [ AreaSeriesService , RangeAreaSeriesService,
  StepAreaSeriesService, StackingAreaSeriesService,
    DateTimeService, CategoryService,
    MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
    eriesService],

```

```

standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='RangeArea' xName='x'
high='high' low='low' name='India' fill='#69D2E7' opacity=0.6></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = rangeData;
    this.primaryXAxis = {
      title: 'Month', valueType: 'Category',
      edgeLabelPlacement: 'Shift'
    };
    this.primaryYAxis = {
      title: 'Temperature (Celsius)',
      minimum: 0, maximum: 20
    };
    this.title = 'Maximum and Minimum Temperature'
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

The following properties can be used to customize the **range area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeAreaSeriesService, StepAreaSeriesService,
StackingAreaSeriesService,
  DateTimeService, CategoryService, MultiColoredAreaSeriesService,
StackingStepAreaSeriesService, SplineRangeAreaSeriesService } from
 '@syncfusion/ej2-angular-charts'

```

```

import { Component, OnInit } from '@angular/core';
import { rangeData } from './datasource';
@Component({
  imports: [
    ChartModule, ChartAllModule
  ],
  providers: [ AreaSeriesService , RangeAreaSeriesService,
    StepAreaSeriesService, StackingAreaSeriesService,
    DateTimeService, CategoryService,
    MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
    eriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
      <e-series-collection>
        <e-series [dataSource]='chartData' type='RangeArea' xName='x'
        high='high' low='low' name='India' fill='#69D2E7' opacity=0.6
        dashArray='5.5' [border]='border'></e-series>
      </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public border?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = rangeData;
    this.border = {
      width: 2,
      color: 'blue'
    }
    this.primaryXAxis = {
      title: 'Month', valueType: 'Category',
      edgeLabelPlacement: 'Shift'
    };
    this.primaryYAxis = {
      title: 'Temperature(Celsius)',
      minimum: 0, maximum: 20
    };
    this.title = 'Maximum and Minimum Temperature'
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

[See Also](#)

- [Data label](#)
- [Tooltip](#)

Range step area in Angular Chart component

[Range step area](#)

To render a range step area series, use series [type](#) as `RangeArea` and inject `RangeAreaSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeAreaSeriesService, StepAreaSeriesService,
  StackingAreaSeriesService,
  DateTimeService, CategoryService, MultiColoredAreaSeriesService,
  StackingStepAreaSeriesService, SplineRangeAreaSeriesService,
  RangeStepAreaSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule, ChartAllModule
  ],
  providers: [ AreaSeriesService , RangeAreaSeriesService,
  StepAreaSeriesService, StackingAreaSeriesService,
  RangeStepAreaSeriesService,
    DateTimeService, CategoryService,
    MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
    eriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='RangeStepArea'
xName='x' high='high' low='low' name='India' opacity=0.4></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData: Object[] = [
    { x: 'Jan', high: 14, low: 4, high1: 29, low1: 19 },
    { x: 'Feb', high: 17, low: 7, high1: 32, low1: 22 },
    { x: 'Mar', high: 20, low: 10, high1: 35, low1: 25 },
    { x: 'Apr', high: 22, low: 12, high1: 37, low1: 27 },
    { x: 'May', high: 20, low: 10, high1: 35, low1: 25 },
    { x: 'Jun', high: 17, low: 7, high1: 32, low1: 22 },
    { x: 'Jul', high: 15, low: 5, high1: 30, low1: 20 },
    { x: 'Aug', high: 17, low: 7, high1: 32, low1: 22 },
    { x: 'Sep', high: 20, low: 10, high1: 35, low1: 25 },
    { x: 'Oct', high: 22, low: 12, high1: 37, low1: 27 },
    { x: 'Nov', high: 20, low: 10, high1: 35, low1: 25 },
```

```

        { x: 'Dec', high: 17, low: 7, high1: 32, low1: 22 }
    ];
    public title?: string;
    public primaryYAxis?: Object;
    ngOnInit(): void {
        this.primaryXAxis = {
            valueType: 'Category',
            edgeLabelPlacement: 'Shift',
            majorGridLines: { width: 0 }
        };
        this.primaryYAxis = {
            labelFormat: '{value}°C',
            lineStyle: { width: 0 },
            minimum: 0,
            maximum: 40,
            majorTickLines: { width: 0 }
        };
        this.title = 'Monthly Temperature Range'
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

The following properties can be used to customize the **range step area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [step](#) – Specifies the position of the step for the series.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeStepAreaSeriesService,
StepAreaSeriesService, StackingAreaSeriesService,
    DateTimeService, CategoryService, MultiColoredAreaSeriesService,
StackingStepAreaSeriesService, SplineRangeAreaSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
    imports: [
        ChartModule, ChartAllModule
    ],
    providers: [ AreaSeriesService , RangeStepAreaSeriesService,
StepAreaSeriesService, StackingAreaSeriesService,
RangeStepAreaSeriesService,

```

```

        DateTimeService, CategoryService,
        MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
        eriesService],
        standalone: true,
        selector: 'app-container',
        template: `<ejs-chart id="chart-container"
        [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
            <e-series-collection>
                <e-series [dataSource]='chartData' type='RangeStepArea'
                xName='x' high='high' low='low' opacity=0.4 fill='red' [border]='border'
                step='Center' dashArray='5,5'></e-series>
            </e-series-collection>
        </ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData: Object[] = [
            { x: 'Jan', high: 14, low: 7, high1: 29, low1: 19 },
            { x: 'Feb', high: 17, low: 7, high1: 32, low1: 22 },
            { x: 'Mar', high: 20, low: 10, high1: 35, low1: 25 },
            { x: 'Apr', high: 22, low: 12, high1: 37, low1: 27 },
            { x: 'May', high: 20, low: 10, high1: 35, low1: 25 },
            { x: 'Jun', high: 17, low: 7, high1: 32, low1: 22 },
            { x: 'Jul', high: 15, low: 5, high1: 30, low1: 20 },
            { x: 'Aug', high: 17, low: 7, high1: 32, low1: 22 },
            { x: 'Sep', high: 20, low: 10, high1: 35, low1: 25 },
            { x: 'Oct', high: 22, low: 12, high1: 37, low1: 27 },
            { x: 'Nov', high: 20, low: 10, high1: 35, low1: 25 },
            { x: 'Dec', high: 20, low: 7, high1: 32, low1: 22 }
        ];
        public title?: string;
        public primaryYAxis?: Object;
        public border?: Object;
        ngOnInit(): void {
            this.primaryXAxis = {
                valueType: 'Category',
                edgeLabelPlacement: 'Shift',
                majorGridLines: { width: 0 }
            };
            this.primaryYAxis = {
                labelFormat: '{value}°C',
                lineStyle: { width: 0 },
                minimum: 0,
                maximum: 40,
                majorTickLines: { width: 0 }
            };
            this.border={
                width: 2,
                color: 'black'
            };
            this.title = 'Monthly Temperature Range'
        }
    }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See also

- [Data label](#)
- [Tooltip](#)

Spline Range Area in Angular Chart component

Spline Range Area

The Spline Range Area Chart is used to display continuous data points as a set of splines that vary between high and low values over intervals of time and across different categories.

To render a spline range area series, use series [type](#) as `SplineRangeArea` and inject `SplineRangeAreaSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeAreaSeriesService, StepAreaSeriesService,
StackingAreaSeriesService,
    DateTimeService, CategoryService, MultiColoredAreaSeriesService,
StackingStepAreaSeriesService, SplineRangeAreaSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { splineRangeData } from './datasource';
@Component({
  imports: [
    ChartModule, ChartAllModule
  ],
  providers: [ AreaSeriesService , RangeAreaSeriesService,
    StepAreaSeriesService, StackingAreaSeriesService,
    DateTimeService, CategoryService,
    MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
    eriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='SplineRangeArea'
xName='x' high='high' low='low' name='England' opacity=0.4></e-series>
      <e-series [dataSource]='chartData' type='SplineRangeArea'
xName='x' high='high1' low='low1' name='India' opacity=0.4></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
```

```

public title?: string;
public primaryYAxis?: Object;
ngOnInit(): void {
    this.chartData = splineRangeData;
    this.primaryXAxis = {
        valueType: 'Category',
        edgeLabelPlacement: 'Shift',
        majorGridLines: { width: 0 }
    };
    this.primaryYAxis = {
        labelFormat: '{value}°C',
        lineStyle: { width: 0 },
        minimum: 0,
        maximum: 40,
        majorTickLines: { width: 0 }
    };
    this.title = 'Monthly Temperature Range'
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

The following properties can be used to customize the **spline range area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeAreaSeriesService, StepAreaSeriesService,
StackingAreaSeriesService,
    DateTimeService, CategoryService, MultiColoredAreaSeriesService,
StackingStepAreaSeriesService, SplineRangeAreaSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { splineRangeData } from './datasource';
@Component({
    imports: [
        ChartModule, ChartAllModule
    ],
    providers: [ AreaSeriesService , RangeAreaSeriesService,
StepAreaSeriesService, StackingAreaSeriesService,

```



```

        DateTimeService, CategoryService,
        MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
        eriesService],
        standalone: true,
        selector: 'app-container',
        template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'>
        <e-series-collection>
            <e-series [dataSource]='chartData' type='SplineRangeArea'
xName='x' high='high' low='low' name='England' opacity=0.4 fill='red'
[border]='border'></e-series>
            <e-series [dataSource]='chartData' type='SplineRangeArea'
xName='x' high='high1' low='low1' name='India' opacity=0.4 fill='blue'
[border]='border'></e-series>
        </e-series-collection>
    </ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public border?: Object;
        public chartData?: Object[];
        public title?: string;
        public primaryYAxis?: Object;
        ngOnInit(): void {
            this.chartData = splineRangeData;
            this.border = {
                width: 2,
                color: 'brown'
            }
            this.primaryXAxis = {
                valueType: 'Category',
                edgeLabelPlacement: 'Shift',
                majorGridLines: { width: 0 }
            };
            this.primaryYAxis = {
                labelFormat: '{value}°C',
                lineStyle: { width: 0 },
                minimum: 0,
                maximum: 40,
                majorTickLines: { width: 0 }
            };
            this.title = 'Monthly Temperature Range'
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Data label](#)

- [Tooltip](#)

Stacked Area in Angular Chart component

Stacked Area

To render a stacked area series, use series [type](#) as `StackingArea` and inject `StackingAreaSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeAreaSeriesService, StepAreaSeriesService,
  StackingAreaSeriesService,
  DateTimeService, CategoryService, MultiColoredAreaSeriesService,
  StackingStepAreaSeriesService, SplineRangeAreaSeriesService } from
  '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stackedData } from './datasource';
@Component({
  imports: [
    ChartModule, ChartAllModule
  ],
  providers: [ AreaSeriesService , RangeAreaSeriesService,
    StepAreaSeriesService, StackingAreaSeriesService,
    DateTimeService, CategoryService,
    MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
    eriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
      <e-series-collection>
        <e-series [dataSource]='chartData' type='StackingArea' xName='x'
yName='y' name='Organic'></e-series>
        <e-series [dataSource]='chartData' type='StackingArea' xName='x'
yName='y1' name='Fair-trade'></e-series>
        <e-series [dataSource]='chartData' type='StackingArea' xName='x'
yName='y2' name='Veg Alternatives'></e-series>
        <e-series [dataSource]='chartData' type='StackingArea' xName='x'
yName='y3' name='Others'></e-series>
      </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = stackedData;
    this.primaryXAxis = {
      valueType: 'DateTime'
    };
    this.title = 'Trend in Sales of Ethical Produce';
  }
}
```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Series customization

The following properties can be used to customize the **stacked area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeAreaSeriesService, StepAreaSeriesService,
StackingAreaSeriesService,
    DateTimeService, CategoryService, MultiColoredAreaSeriesService,
StackingStepAreaSeriesService, SplineRangeAreaSeriesService } from
'@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stackedData } from './datasource';
@Component({
  imports: [
    ChartModule, ChartAllModule
  ],
  providers: [ AreaSeriesService , RangeAreaSeriesService,
StepAreaSeriesService, StackingAreaSeriesService,
    DateTimeService, CategoryService,
MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
eriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='StackingArea' xName='x'
yName='y' name='Organic' [border]='border' fill= 'red'></e-series>
      <e-series [dataSource]='chartData' type='StackingArea' xName='x'
yName='y1' name='Fair-trade' [border]='border' fill='yellow'></e-series>
      <e-series [dataSource]='chartData' type='StackingArea' xName='x'
yName='y2' name='Veg Alternatives' [border]='border' fill='green'></e-
series>
      <e-series [dataSource]='chartData' type='StackingArea' xName='x'
yName='y3' name='Others' [border]='border' fill='blue'></e-series>
    </e-series-collection>
  </ejs-chart>`
```

```

    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public border?: Object;
        public chartData?: Object[];
        public title?: string;
        public primaryYAxis?: Object;
        ngOnInit(): void {
            this.chartData = stackedData;
            this.border = {
                width: 2.5,
                color: 'white'
            }
            this.primaryXAxis = {
                valueType: 'DateTime'
            };
            this.title = 'Trend in Sales of Ethical Produce';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Data label](#)
- [Tooltip](#)

100% Stacked Area in Angular Chart component

100% Stacked Area

To render a 100% stacked area series, use series [type](#) as `StackingArea100` and inject `StackingAreaSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeAreaSeriesService, StepAreaSeriesService,
    StackingAreaSeriesService,
    DateTimeService, CategoryService, MultiColoredAreaSeriesService,
    StackingStepAreaSeriesService, SplineRangeAreaSeriesService } from
    '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { percentData } from './datasource';
@Component({
    imports: [
        ChartModule, ChartAllModule
    ],

```

```

providers: [ AreaSeriesService , RangeAreaSeriesService,
StepAreaSeriesService, StackingAreaSeriesService,
              DateTimeService, CategoryService,
MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
seriesService],
standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='StackingArea100'
xName='x' yName='y' name='USA'></e-series>
      <e-series [dataSource]='chartData' type='StackingArea100'
xName='x' yName='y1' name='UK'></e-series>
      <e-series [dataSource]='chartData' type='StackingArea100'
xName='x' yName='y2' name='Canada'></e-series>
      <e-series [dataSource]='chartData' type='StackingArea100'
xName='x' yName='y3' name='China'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = percentData;
    this.primaryXAxis = {
      valueType: 'DateTime'
    };
    this.title = 'Annual Temperature Comparison';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

The following properties can be used to customize the **100% stacked area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'

```

```

import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeAreaSeriesService, StepAreaSeriesService,
StackingAreaSeriesService,
    DateTimeService, CategoryService, MultiColoredAreaSeriesService,
StackingStepAreaSeriesService, SplineRangeAreaSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { percentData } from './datasource';
@Component({
imports: [
    ChartModule, ChartAllModule
],
providers: [ AreaSeriesService , RangeAreaSeriesService,
StepAreaSeriesService, StackingAreaSeriesService,
    DateTimeService, CategoryService,
MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
eriesService],
standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='StackingArea100'
xName='x' yName='y' name='USA' [border] = 'border' fill='red'></e-series>
        <e-series [dataSource]='chartData' type='StackingArea100'
xName='x' yName='y1' name='UK' [border] = 'border' fill='yellow'></e-series>
        <e-series [dataSource]='chartData' type='StackingArea100'
xName='x' yName='y2' name='Canada' [border] = 'border' fill='green'></e-
series>
        <e-series [dataSource]='chartData' type='StackingArea100'
xName='x' yName='y3' name='China' [border] = 'border' fill='blue'></e-
series>
    </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public border?: Object;
    public title?: string;
    public primaryYAxis?: Object;
    ngOnInit(): void {
        this.chartData = percentData;
        this.border = {
            width: 2.5,
            color: 'white'
        }
        this.primaryXAxis = {
            valueType: 'DateTime'
        };
        this.title = 'Annual Temperature Comparison';
    }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Data label](#)
- [Tooltip](#)

Stacked step area in Angular Chart component

Stacked step area

To render a Stacked Step Area series, use series [type](#) as `StackingStepArea` and inject `StackingStepAreaSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeAreaSeriesService, StepAreaSeriesService,
StackingAreaSeriesService,
    DateTimeService, CategoryService, MultiColoredAreaSeriesService,
StackingStepAreaSeriesService, SplineRangeAreaSeriesService } from
'@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { percentData } from './datasource';
@Component({
imports: [
    ChartModule, ChartAllModule
],
providers: [ AreaSeriesService , RangeAreaSeriesService,
StepAreaSeriesService, StackingAreaSeriesService,
    DateTimeService, CategoryService,
MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
eriesService],
standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='StackingStepArea'
xName='x' yName='y' name='USA'></e-series>
        <e-series [dataSource]='chartData' type='StackingStepArea'
xName='x' yName='y1' name='UK'></e-series>
    </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    ngOnInit(): void {
```

```

        this.chartData = percentData;
        this.primaryXAxis = {
            valueType: 'DateTime'
        };
        this.title = 'Annual Temperature Comparison';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

The following properties can be used to customize the **Stacked Step Area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [step](#) – Specifies the position of the step for the series.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeAreaSeriesService, StepAreaSeriesService,
    StackingAreaSeriesService,
    DateTimeService, CategoryService, MultiColoredAreaSeriesService,
    StackingStepAreaSeriesService, SplineRangeAreaSeriesService } from
    '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { percentData } from './datasource';
@Component({
    imports: [
        ChartModule, ChartAllModule
    ],
    providers: [ AreaSeriesService , RangeAreaSeriesService,
        StepAreaSeriesService, StackingAreaSeriesService,
        DateTimeService, CategoryService,
        MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
        eriesService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
        <e-series-collection>
            <e-series [dataSource]='chartData' type='StackingStepArea'
            xName='x' yName='y' name='USA' [border]='border' dashArray='5.5'
            step='Center' opacity=0.5 fill='red'></e-series>

```



```

        <e-series [dataSource]='chartData' type='StackingStepArea'
xName='x' yName='y1' name='UK' [border]='border' dashArray='5.5'
step='Center' opacity=0.5 fill='green'></e-series>
    </e-series-collection>
</ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public primaryYAxis?: Object;
        public border?: Object;
        ngOnInit(): void {
            this.chartData = percentData;
            this.primaryXAxis = {
                valueType: 'DateTime'
            };
            this.border={ width: 2, color: 'yellow' };
            this.title = 'Annual Temperature Comparison';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See also

- [Data label](#)
- [Tooltip](#)

Step area in Angular Chart component

[Step area](#)

To render a step area series, use series [type](#) as `StepArea` and inject `StepAreaSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeAreaSeriesService, StepAreaSeriesService,
StackingAreaSeriesService,
    DateTimeService, CategoryService, MultiColoredAreaSeriesService,
StackingStepAreaSeriesService, SplineRangeAreaSeriesService } from
'@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stepData } from './datasource';
@Component({
    imports: [
        ChartModule, ChartAllModule
    ]
})

```

```

    ],
    providers: [ AreaSeriesService , RangeAreaSeriesService,
    StepAreaSeriesService, StackingAreaSeriesService,
    DateTimeService, CategoryService,
    MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
    eriesService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
    <e-series [dataSource]='chartData' type='StepArea' xName='x'
    yName='y' name='England'></e-series>
    </e-series-collection>
    </ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    ngOnInit(): void {
      this.chartData = stepData;
      this.primaryXAxis = {
        valueType: 'Double',
        title: 'Overs'
      };
      this.primaryYAxis = {
        title: 'Runs'
      };
      this.title = 'England - Run Rate';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

The following properties can be used to customize the **step area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.
- [step](#) – Specifies the position of the step for the series.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'

```

```

import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, RangeAreaSeriesService, StepAreaSeriesService,
StackingAreaSeriesService,
    DateTimeService, CategoryService, MultiColoredAreaSeriesService,
StackingStepAreaSeriesService, SplineRangeAreaSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stepData } from './datasource';
@Component({
imports: [
    ChartModule, ChartAllModule
],
providers: [ AreaSeriesService , RangeAreaSeriesService,
StepAreaSeriesService, StackingAreaSeriesService,
    DateTimeService, CategoryService,
MultiColoredAreaSeriesService, StackingStepAreaSeriesService, SplineRangeAreaS
eriesService],
standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='StepArea' xName='x'
yName='y' name='England' [border]='border' opacity=0.4 fill='red'
step='Right' dashArray='5,5'></e-series>
    </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public border?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    ngOnInit(): void {
        this.chartData = stepData;
        this.border = {
            width: 2,
            color: 'brown'
        }
        this.primaryXAxis = {
            valueType: 'Double',
            title: 'Overs'
        };
        this.primaryYAxis = {
            title: 'Runs'
        };
        this.title = 'England - Run Rate';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';

```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See also

- [Data label](#)
- [Tooltip](#)

Spline Area in Angular Chart component

Spline Area

To render a spline series, use series [type](#) as **Spline** and

inject **SplineAreaSeriesService** into the **@NgModule.providers**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, StepLineSeriesService,
  SplineSeriesService, StackingLineSeriesService, DateTimeService,
  SplineAreaSeriesService, MultiColoredLineSeriesService,
  ParetoSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { splineData } from '../datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService, StepLineSeriesService,
    SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
    ParetoSeriesService, ColumnSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='SplineArea' xName='x'
yName='y' name='London' width=2 [marker]='marker'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.chartData = splineData;
    this.primaryXAxis = {
      title: 'Month',
      valueType: 'Category'
    };
  }
}
```

```

        this.marker = { visible: true, width: 10, height: 10 };
        this.title = 'Climate Graph-2012';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

The following properties can be used to customize the **spline area** series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, StepLineSeriesService,
    SplineSeriesService, StackingLineSeriesService, DateTimeService,
        SplineAreaSeriesService, MultiColoredLineSeriesService,
    ParetoSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { splineData } from './datasource';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ CategoryService, LineSeriesService, StepLineSeriesService,
    SplineSeriesService, StackingLineSeriesService, DateTimeService,
        SplineAreaSeriesService, MultiColoredLineSeriesService,
    ParetoSeriesService, ColumnSeriesService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[title]='title'>
        <e-series-collection>
            <e-series [dataSource]='chartData' type='SplineArea' xName='x'
yName='y' name='London' width=2 [marker]='marker' fill='yellow'
[border]='border'></e-series>
        </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
}

```

```

public title?: string;
public primaryYAxis?: Object;
public marker?: Object;
border?: Object;
ngOnInit(): void {
    this.chartData = splineData;
    this.border = {
        width: 2.5,
        color: 'red'
    }
    this.primaryXAxis = {
        title: 'Month',
        valueType: 'Category'
    };
    this.marker = { visible: false, width: 10, height: 10 };
    this.title = 'Climate Graph-2012';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Data label](#)
- [Tooltip](#)

Column in Angular Chart component

Column

To render a column series, use series [type](#) as `Column` and inject `ColumnSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
ColumnSeriesService, LineSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { columnData } from './datasource';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ CategoryService, DateTimeService, ScrollBarService,
LineSeriesService, ColumnSeriesService,

```

```

        ChartAnnotationService, RangeColumnSeriesService,
        StackingColumnSeriesService, LegendService, TooltipService,],
        standalone: true,
        selector: 'app-container',
        template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis' [title]='title'>
            <e-series-collection>
                <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold'></e-series>
            </e-series-collection>
        </ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        primaryYAxis: any;
        ngOnInit(): void {
            this.chartData = columnData;
            this.primaryXAxis = {
                valueType: 'Category',
                title: 'Countries'
            };
            this.title = 'Olympic Medals';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Column spacing and width

The [columnSpacing](#) and [columnWidth](#) properties are used to customize the space between columns.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
ColumnSeriesService, LineSeriesService,
        ChartAnnotationService, RangeColumnSeriesService,
        StackingColumnSeriesService, LegendService, TooltipService
    } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { columnData } from './datasource';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ CategoryService, DateTimeService, ScrollBarService,
        LineSeriesService, ColumnSeriesService,

```

```

        ChartAnnotationService, RangeColumnSeriesService,
        StackingColumnSeriesService, LegendService, TooltipService,],
        standalone: true,
        selector: 'app-container',
        template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis' [title]='title'>
        <e-series-collection>
            <e-series [dataSource]='chartData' type='Column'
columnSpacing='0.5' xName='country' yName='gold'></e-series>
            <e-series [dataSource]='chartData' type='Column' xName='country'
columnWidth='0.25' yName='silver'></e-series>
        </e-series-collection>
    </ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        primaryYAxis: any;
        ngOnInit(): void {
            this.chartData = columnData;
            this.primaryXAxis = {
                valueType: 'Category',
                title: 'Countries'
            };
            this.title = 'Olympic Medals';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Grouped column

<!-- markdownlint-disable MD010 -->

You can use the [groupName](#) property to group the data points in the column type charts. Data points with same group name are grouped together.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
        ColumnSeriesService, LineSeriesService,
        ChartAnnotationService, RangeColumnSeriesService,
        StackingColumnSeriesService, LegendService, TooltipService
    } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
    imports: [
        ChartModule
    ]
})

```



```

    ],
    providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService, ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart style='display:block;'
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'>
      <e-series-collection>
        <e-series groupName="A" [dataSource]='data' type='Column'
        xName='x' yName='y' name='UK' width='2'> </e-series>
        <e-series groupName="B" [dataSource]='data1'
        type='Column' columnWidth='0.6' xName='x' yName='y' name='Germany'
        width='2'> </e-series>
        <e-series groupName="A" [dataSource]='data2'
        type='Column' xName='x' yName='y' name='France' width='2'> </e-series>
        <e-series groupName="B" [dataSource]='data3'
        type='Column' xName='x' columnWidth='0.6' yName='y' name='Italy' width='2'>
        </e-series>
      </e-series-collection>
    </ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public data?: Object[];
    public data1?: Object[];
    public data2?: Object[];
    public data3?: Object[];
    public primaryXAxis?: Object;
    public primaryYAxis?: Object;
    ngOnInit(): void {
      this.primaryXAxis = {
        majorGridLines: { width: 0 },
        minorGridLines: { width: 0 },
        majorTickLines: { width: 0 },
        minorTickLines: { width: 0 },
        interval: 1,
        lineStyle: { width: 0 },
        valueType: "Category"
      };
      this.primaryYAxis = {
        title: "Sales",
        lineStyle: { width: 0 },
        majorTickLines: { width: 0 },
        majorGridLines: { width: 1 },
        minorGridLines: { width: 1 },
        minorTickLines: { width: 0 },
        labelFormat: "{value}B"
      };
      this.data = [
        { x: "2014", y: 111.1 },
        { x: "2015", y: 127.3 },
        { x: "2016", y: 143.4 },
        { x: "2017", y: 159.9 }
      ];
      this.data1 = [
        { x: "2014", y: 76.9 },

```

```

    { x: "2015", y: 99.5 },
    { x: "2016", y: 121.7 },
    { x: "2017", y: 142.5 }
  ];
  this.data2 = [
    { x: "2014", y: 66.1 },
    { x: "2015", y: 79.3 },
    { x: "2016", y: 91.3 },
    { x: "2017", y: 102.4 }
  ];
  this.data3 = [
    { x: "2014", y: 34.1 },
    { x: "2015", y: 38.2 },
    { x: "2016", y: 44.0 },
    { x: "2017", y: 51.6 }
  ];
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Cylindrical column chart

To render a cylindrical column chart, set the [columnFacet](#) property to **Cylinder** in the chart series.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
  ColumnSeriesService, LineSeriesService,
  ChartAnnotationService, RangeColumnSeriesService,
  StackingColumnSeriesService, LegendService, TooltipService
  } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { cylindricalData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService, ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis' [title]='title' [tooltip]='tooltip'>
    <e-series-collection>

```

```

        <e-series [dataSource]='chartData' type='Column' columnFacet=
        'Cylinder' xName='country' yName='gold'
        tooltipMappingName='tooltipMappingName'></e-series>
    </e-series-collection>
</ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public primaryYAxis?: Object;
        public tooltip?: Object;
        ngOnInit(): void {
            this.chartData = cylindricalData;
            this.primaryXAxis = {
                valueType: 'Category',
                interval: 1
            };
            this.primaryYAxis = {
                minimum: 0,
                maximum: 80,
                interval: 10,
                title: 'Medal Count'
            };
            this.title = 'Olympic Gold Medal Counts - RIO';
            this.tooltip = { enable: true, header: "<b>${point.tooltip}</b>",
format: "Gold Medal: <b>${point.y}</b>" };
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

The following properties can be used to customize the **column** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
ColumnSeriesService, LineSeriesService,
ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService

```

```

    } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { columnData } from '../datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService, ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column'
columnWidth='0.5' columnSpacing='0.5' xName='country' yName='gold'
fill='yellow' [border]='border' dashArray='5.5' opacity='0.5'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public border?: Object;
  primaryYAxis: any;
  ngOnInit(): void {
    this.border = {
      color: 'green',
      width: 2
    };
    this.chartData = columnData;
    this.primaryXAxis = {
      valueType: 'Category',
      title: 'Countries'
    };
    this.title = 'Olympic Medals';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See also

- [Data label](#)
- [Tooltip](#)

Range Column in Angular Chart component

Range Column

To render a range column series, use series [type](#) as `RangeColumn` and inject `RangeColumnSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
ColumnSeriesService, LineSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { data1 } from './datasource';
import { data2 } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService, ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='data1' type='RangeColumn' xName='x'
low='low' high='high'></e-series>
      <e-series [dataSource]='data2' type='RangeColumn' xName='x'
low='low' high='high'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public data1?: Object[];
  public data2?: Object[];
  primaryYAxis: any;
  ngOnInit(): void {
    this.data1 = data1;
    this.data2 = data2;
    this.primaryXAxis = {
      title: 'month',
      valueType: 'Category'
    };
    this.title = 'Maximum and minimum Temperature';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Series customization

The following properties can be used to customize the **range column** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
ColumnSeriesService, LineSeriesService,
ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { data1 } from './datasource';
import { data2 } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
LineSeriesService, ColumnSeriesService,
ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService, ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[title]='title'>
    <e-series-collection>
      <e-series [dataSource]='data1' type='RangeColumn' xName='x'
low='low' high='high' fill='yellow' [border]='border' dashArray='5.5'></e-
series>
      <e-series [dataSource]='data2' type='RangeColumn' xName='x'
low='low' high='high' fill='yellow' [border]='border' dashArray='5.5'></e-
series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public border?: Object;
  public title?: string;
  public data1?: Object[];
```

```

public data2?: Object[];
ngOnInit(): void {
  this.data1 = data1;
  this.data2 = data2;
  this.border = {
    width: 2.5,
    color: 'brown'
  }
  this.primaryXAxis = {
    title: 'month',
    valueType: 'Category'
  };
  this.title = 'Maximum and minimum Temperature';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Data label](#)
- [Tooltip](#)

Stacked Column in Angular Chart component

Stacked column

To render a stacked column series, use series [type](#) as `StackingColumn` and inject `StackingColumnSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
  ColumnSeriesService, LineSeriesService,
  ChartAnnotationService, RangeColumnSeriesService,
  StackingColumnSeriesService, LegendService, TooltipService
  } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stackedData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService, ],
  standalone: true,

```

```

        selector: 'app-container',
        template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
        <e-series-collection>
            <e-series [dataSource]='chartData' type='StackingColumn'
xName='x' yName='y' name='UK'></e-series>
            <e-series [dataSource]='chartData' type='StackingColumn'
xName='x' yName='y1' name='Germany'></e-series>
            <e-series [dataSource]='chartData' type='StackingColumn'
xName='x' yName='y2' name='France'></e-series>
            <e-series [dataSource]='chartData' type='StackingColumn'
xName='x' yName='y3' name='Italy'></e-series>
        </e-series-collection>
    </ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        primaryYAxis: any;
        ngOnInit(): void {
            this.chartData = stackedData;
            this.primaryXAxis = {
                title: 'Years',
                interval: 1,
                valueType: 'Category'
            };
            this.title = 'Mobile Game Market by Country';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Stacking group

<!-- markdownlint-disable MD010 -->

You can use the [stackingGroup](#) property to group the stacked columns and 100% stacked columns.

Columns with same group name are stacked on top of each other.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
ColumnSeriesService, LineSeriesService,
ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';

```



```

@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService, ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart style='display:block;'
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'>
      <e-series-collection>
        <e-series stackingGroup="A" [dataSource]='data'
type='StackingColumn' xName='x' yName='y' name='UK' width='2'> </e-series>
        <e-series stackingGroup="B" [dataSource]='data1'
type='StackingColumn' xName='x' yName='y' name='Germany' width='2'> </e-
series>
        <e-series stackingGroup="A" [dataSource]='data2'
type='StackingColumn' xName='x' yName='y' name='France' width='2'> </e-
series>
        <e-series stackingGroup="B" [dataSource]='data3'
type='StackingColumn' xName='x' yName='y' name='Italy' width='2'> </e-
series>
      </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public data1?: Object[];
  public data2?: Object[];
  public data3?: Object[];
  public primaryXAxis?: Object;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.primaryXAxis = {
      majorGridLines: { width: 0 },
      minorGridLines: { width: 0 },
      majorTickLines: { width: 0 },
      minorTickLines: { width: 0 },
      interval: 1,
      lineStyle: { width: 0 },
      valueType: "Category"
    };
    this.primaryYAxis = {
      title: "Sales",
      lineStyle: { width: 0 },
      majorTickLines: { width: 0 },
      majorGridLines: { width: 1 },
      minorGridLines: { width: 1 },
      minorTickLines: { width: 0 },
      labelFormat: "{value}B"
    };
    this.data = [
      { x: "2014", y: 111.1 },
      { x: "2015", y: 127.3 },
      { x: "2016", y: 143.4 },
    ]
  }
}

```

```

    { x: "2017", y: 159.9 }
  ];
  this.data1 = [
    { x: "2014", y: 76.9 },
    { x: "2015", y: 99.5 },
    { x: "2016", y: 121.7 },
    { x: "2017", y: 142.5 }
  ];
  this.data2 = [
    { x: "2014", y: 66.1 },
    { x: "2015", y: 79.3 },
    { x: "2016", y: 91.3 },
    { x: "2017", y: 102.4 }
  ];
  this.data3 = [
    { x: "2014", y: 34.1 },
    { x: "2015", y: 38.2 },
    { x: "2016", y: 44.0 },
    { x: "2017", y: 51.6 }
  ];
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Cylindrical stacked column chart

To render a cylindrical stacked column chart, set the `columnFacet` property to `Cylinder` in the chart series.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
  ColumnSeriesService, LineSeriesService,
  ChartAnnotationService, RangeColumnSeriesService,
  StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stackedData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService, ],
  standalone: true,

```

```

    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='StackingColumn'
columnFacet= 'Cylinder' xName='x' yName='y' name='UK'></e-series>
        <e-series [dataSource]='chartData' type='StackingColumn'
columnFacet= 'Cylinder' xName='x' yName='y1' name='Germany'></e-series>
        <e-series [dataSource]='chartData' type='StackingColumn'
columnFacet= 'Cylinder' xName='x' yName='y2' name='France'></e-series>
        <e-series [dataSource]='chartData' type='StackingColumn'
columnFacet= 'Cylinder' xName='x' yName='y3' name='Italy'></e-series>
    </e-series-collection>
    </ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public primaryYAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    ngOnInit(): void {
      this.chartData = stackedData;
      this.primaryXAxis = {
        title: 'Years',
        interval: 1,
        valueType: 'Category'
      };
      this.primaryYAxis = {
        title: 'Sales in Billions',
        minimum: 0,
        maximum: 700,
        interval: 100,
        labelFormat: '{value}B',
      };
      this.title = 'Mobile Game Market by Country';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

The following properties can be used to customize the **stacked column** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
ColumnSeriesService, LineSeriesService,
ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stackedData } from './datasource';
@Component({
imports: [
ChartModule
],
providers: [ CategoryService, DateTimeService, ScrollBarService,
LineSeriesService, ColumnSeriesService,
ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService, ],
standalone: true,
selector: 'app-container',
template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
<e-series-collection>
<e-series [dataSource]='chartData' type='StackingColumn'
xName='x' yName='y' name='UK' fill='red' [border]='border'></e-series>
<e-series [dataSource]='chartData' type='StackingColumn'
xName='x' yName='y1' name='Germany' fill='yellow' [border]='border'></e-
series>
<e-series [dataSource]='chartData' type='StackingColumn'
xName='x' yName='y2' name='France' fill='green' [border]='border'></e-
series>
<e-series [dataSource]='chartData' type='StackingColumn'
xName='x' yName='y3' name='Italy' fill='blue' [border]='border'></e-series>
</e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
public primaryXAxis?: Object;
public border?: Object;
public chartData?: Object[];
public title?: string;
primaryYAxis: any;
ngOnInit(): void {
this.chartData = stackedData;
this.border = {
width: 2.5,
color: 'blue'
}
this.primaryXAxis = {
title: 'Years',
interval: 1,
valueType: 'Category'
};
this.title = 'Mobile Game Market by Country';
}
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See also

- [Data label](#)
- [Tooltip](#)

100% Stacked Column in Angular Chart component

100% Stacked column

To render a 100% stacked column series, use series [type](#) as **StackingColumn100** and inject **StackingColumnSeriesService** into the **@NgModule.providers**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
ColumnSeriesService, LineSeriesService,
ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { percentData } from './datasource';
@Component({
imports: [
ChartModule
],
providers: [ CategoryService, DateTimeService, ScrollBarService,
LineSeriesService, ColumnSeriesService,
ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService, ],
standalone: true,
selector: 'app-container',
template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
<e-series-collection>
<e-series [dataSource]='chartData' type='StackingColumn100'
xName='x' yName='y' name='UK'></e-series>
<e-series [dataSource]='chartData' type='StackingColumn100'
xName='x' yName='y1' name='Germany'></e-series>
<e-series [dataSource]='chartData' type='StackingColumn100'
xName='x' yName='y2' name='France'></e-series>
<e-series [dataSource]='chartData' type='StackingColumn100'
xName='x' yName='y3' name='Italy'></e-series>
</e-series-collection>
</ejs-chart>`
})
```

```
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = percentData;
    this.primaryXAxis = {
      title: 'Years',
      interval: 1,
      valueType: 'Category'
    };
    this.title = 'Gross Domestic Product Growth';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

100% Cylindrical stacked column chart

To render a 100% cylindrical stacked column chart, set the [columnFacet](#) property to **Cylinder** in the chart series.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
  ColumnSeriesService, LineSeriesService,
  ChartAnnotationService, RangeColumnSeriesService,
  StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { cylindricalData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService, ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='StackingColumn100'
columnFacet= 'Cylinder' xName='x' yName='y' name='UK'></e-series>
```

```

        <e-series [dataSource]='chartData' type='StackingColumn100'
columnFacet= 'Cylinder' xName='x' yName='y1' name='Germany'></e-series>
        <e-series [dataSource]='chartData' type='StackingColumn100'
columnFacet= 'Cylinder' xName='x' yName='y2' name='France'></e-series>
        <e-series [dataSource]='chartData' type='StackingColumn100'
columnFacet= 'Cylinder' xName='x' yName='y3' name='Italy'></e-series>
    </e-series-collection>
</ejs-chart>`
    ))
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    ngOnInit(): void {
        this.chartData = cylindricalData;
        this.primaryXAxis = {
            title: 'Years',
            interval: 1,
            valueType: 'DateTime',
            labelFormat: 'Y'
        };
        this.primaryYAxis = {
            title: 'GDP (%) Per Annum',
            rangePadding: 'None',
            labelFormat: '{value}%'
        };
        this.title = 'Gross Domestic Product Growth';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

The following properties can be used to customize the 100% stacked column series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
ColumnSeriesService, LineSeriesService,

```

```

    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService
  } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { percentData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService, ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='StackingColumn100'
xName='x' yName='y' name='UK' fill='red' [border]='border'></e-series>
    <e-series [dataSource]='chartData' type='StackingColumn100'
xName='x' yName='y1' name='Germany' fill='yellow' [border]='border'></e-
series>
    <e-series [dataSource]='chartData' type='StackingColumn100'
xName='x' yName='y2' name='France' fill='green' [border]='border'></e-
series>
    <e-series [dataSource]='chartData' type='StackingColumn100'
xName='x' yName='y3' name='Italy' fill='blue' [border]='border'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public border?: Object;
  public title?: string;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = percentData;
    this.border = {
      width: 2.5,
      color: 'brown'
    }
    this.primaryXAxis = {
      title: 'Years',
      interval: 1,
      valueType: 'Category'
    };
    this.title = 'Gross Domestic Product Growth';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';

```



```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See also

- [Data label](#)
- [Tooltip](#)

Bar in Angular Chart component

Bar

To render a bar series, use series [type](#) as **Bar** and inject **BarSeriesService** into the **@NgModule.providers**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { BarSeriesService, StackingBarSeriesService, CategoryService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { barData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ BarSeriesService, StackingBarSeriesService, CategoryService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Bar' xName='x'
yName='y' name='India'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public title?: string;
  primaryXAxis: any;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = barData;
    this.title = 'Unemployment rate (%)';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Bar space and width

The [columnSpacing](#) and [columnWidth](#) properties are used to customize the space between bars.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { BarSeriesService, StackingBarSeriesService, CategoryService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { barData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ BarSeriesService, StackingBarSeriesService, CategoryService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Bar' xName='x'
yName='y' name='India' columnwidth='0.5' columnSpacing='0.5'></e-series>
    <e-series [dataSource]='chartData' type='Bar' xName='x'
yName='y1' name='India'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public title?: string;
  primaryXAxis: any;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = barData;
    this.title = 'Unemployment rate (%)';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Grouped bar

You can use the [groupName](#) property to group the data points in the bar type charts. Data points with same group name are grouped together.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
```

```

import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { BarSeriesService, StackingBarSeriesService, CategoryService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { groupData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ BarSeriesService, StackingBarSeriesService, CategoryService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Bar' xName='x'
yName='y' name='John' groupName='JohnandAndrew'></e-series>
      <e-series [dataSource]='chartData' type='Bar' xName='x'
yName='y1' name='Andrew' groupName='JohnandAndrew' columnWidth='0.6'></e-
series>
      <e-series [dataSource]='chartData' type='Bar' xName='x'
yName='y2' name='Thomas' groupName='ThomasandMichael'></e-series>
      <e-series [dataSource]='chartData' type='Bar' xName='x'
yName='y3' name='Michael' columnWidth='0.6'
groupName='ThomasandMichael'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public title?: string;
  primaryXAxis: any;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = groupData;
    this.title = 'Sales by year';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Cylindrical bar chart

To render a cylindrical bar chart, set the [columnFacet](#) property to **Cylinder** in the chart series.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'

```

```

import { BarSeriesService, StackingBarSeriesService, CategoryService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { cylindricalData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ BarSeriesService, StackingBarSeriesService, CategoryService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Bar'
columnFacet='Cylinder' xName='x' yName='y' name='India'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public title?: string;
  public primaryXAxis?: Object;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = cylindricalData;
    this.title = 'Unemployment rate in percentage';
    this.primaryXAxis = {
      minimum: 2005,
      maximum: 2012,
      interval: 1
    };
    this.primaryYAxis = {
      minimum: 3,
      maximum: 12,
      interval: 1,
      title: 'Percentage'
    };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

The following properties can be used to customize the **bar** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.

- [border](#) – Specifies the [color](#) and [width](#) of series border.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { BarSeriesService, StackingBarSeriesService, CategoryService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { barData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ BarSeriesService, StackingBarSeriesService, CategoryService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Bar' xName='x'
yName='y' name='India' fill='yellow' dashArray='5.5' opacity='0.8'
[border]='border'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public title?: string;
  public border?: Object;
  primaryXAxis: any;
  primaryYAxis: any;
  ngOnInit(): void {
    this.border = {
      color: 'brown',
      width: 1
    };
    this.chartData = barData;
    this.title = 'Unemployment rate (%)';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See also

- [Data label](#)
- [Tooltip](#)

Stacked Bar in Angular Chart component

Stacked bar

To render a stacked bar series, use series [type](#) as `StackingBar` and inject `StackingBarSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { BarSeriesService, StackingBarSeriesService, CategoryService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stackData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ BarSeriesService, StackingBarSeriesService, CategoryService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='StackingBar' xName='x'
yName='y' name='Apple'></e-series>
    <e-series [dataSource]='chartData' type='StackingBar' xName='x'
yName='y1' name='Orange'></e-series>
    <e-series [dataSource]='chartData' type='StackingBar' xName='x'
yName='y2' name='Wastage'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = stackData;
    this.primaryXAxis = {
      valueType: 'Category',
      title: 'Months'
    };
    this.title = 'Sales Comparison';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Cylindrical stacked bar chart

To render a cylindrical stacked bar chart, set the [columnFacet](#) property to **Cylinder** in the chart series.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { BarSeriesService, StackingBarSeriesService, CategoryService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { cylindricalData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ BarSeriesService, StackingBarSeriesService, CategoryService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='StackingBar'
columnFacet= 'Cylinder' xName='x' yName='y'></e-series>
    <e-series [dataSource]='chartData' type='StackingBar'
columnFacet= 'Cylinder' xName='x' yName='y1'></e-series>
    <e-series [dataSource]='chartData' type='StackingBar'
columnFacet= 'Cylinder' xName='x' yName='y2'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  primaryYAxis: any;
  primaryXAxis: any;
  ngOnInit(): void {
    this.chartData = cylindricalData;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Series customization

The following properties can be used to customize the **stacked bar** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { BarSeriesService, StackingBarSeriesService, CategoryService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stackData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ BarSeriesService, StackingBarSeriesService, CategoryService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='StackingBar' xName='x'
yName='y' name='Apple' fill='yellow' [border]='border' dashArray='5.5'></e-
series>
      <e-series [dataSource]='chartData' type='StackingBar' xName='x'
yName='y1' name='Orange' fill='green' [border]='border1'
dashArray='5.5'></e-series>
      <e-series [dataSource]='chartData' type='StackingBar' xName='x'
yName='y2' name='Wastage' fill='blue' [border]='border2'
dashArray='5.5'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public border?: Object;
  public border1?: Object;
  public border2?: Object;
  public chartData?: Object[];
  public title?: string;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = stackData;
    this.border = {
      width: 2,
      color: 'brown'
    };
    this.border1 = {
      width: 2,
      color: 'red'
    };
    this.border2 = {
      width: 2,
      color: 'grey'
    };
    this.primaryXAxis = {
      valueType: 'Category',
      title: 'Months'
    };
  };
}

```



```

        this.title = 'Sales Comparison';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See also

- [Data label](#)
- [Tooltip](#)

100% Stacked Bar in Angular Chart component

100% Stacked bar

To render a 100% stacked bar series, use series [type](#) as `StackingBar100` and inject `StackingBarSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { BarSeriesService, StackingBarSeriesService, CategoryService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stackData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ BarSeriesService, StackingBarSeriesService, CategoryService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='StackingBar100'
xName='x' yName='y' name='Apple'></e-series>
      <e-series [dataSource]='chartData' type='StackingBar100'
xName='x' yName='y1' name='Orange'></e-series>
      <e-series [dataSource]='chartData' type='StackingBar100'
xName='x' yName='y2' name='Wastage'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  primaryYAxis: any;
}

```

```

ngOnInit(): void {
    this.chartData = stackData;
    this.primaryXAxis = {
        valueType: 'Category',
        title: 'Months'
    };
    this.title = 'Sales Comparison';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

100% Cylindrical stacked bar chart

To render a 100% cylindrical stacked bar chart, set the [columnFacet](#) property to **Cylinder** in the chart series.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { BarSeriesService, StackingBarSeriesService, CategoryService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { cylindricalData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ BarSeriesService, StackingBarSeriesService, CategoryService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='StackingBar100'
columnFacet= 'Cylinder' xName='x' yName='y'></e-series>
      <e-series [dataSource]='chartData' type='StackingBar100'
columnFacet= 'Cylinder' xName='x' yName='y1'></e-series>
      <e-series [dataSource]='chartData' type='StackingBar100'
columnFacet= 'Cylinder' xName='x' yName='y2'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  primaryXAxis: any;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = cylindricalData;
  }
}

```

```
}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Series customization

The following properties can be used to customize the **100% stacked bar** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.
- [border](#) – Specifies the [color](#) and [width](#) of series border.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { BarSeriesService, StackingBarSeriesService, CategoryService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stackData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ BarSeriesService, StackingBarSeriesService, CategoryService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='StackingBar100'
xName='x' yName='y' name='Apple' fill='yellow' [border]='border'
dashArray='5.5'></e-series>
    <e-series [dataSource]='chartData' type='StackingBar100'
xName='x' yName='y1' name='Orange' fill='green' [border]='border1'
dashArray='5.5'></e-series>
    <e-series [dataSource]='chartData' type='StackingBar100'
xName='x' yName='y2' name='Wastage' fill='blue' [border]='border2'
dashArray='5.5'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public border?: Object;
  public border1?: Object;
  public border2?: Object;
```

```

public chartData?: Object[];
public title?: string;
primaryYAxis: any;
ngOnInit(): void {
    this.chartData = stackData;
    this.border = {
        width: 2,
        color: 'brown'
    };
    this.border1 = {
        width: 2,
        color: 'red'
    };
    this.border2 = {
        width: 2,
        color: 'grey'
    };
    this.primaryXAxis = {
        valueType: 'Category',
        title: 'Months'
    };
    this.title = 'Sales Comparison';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See also

- [Data label](#)
- [Tooltip](#)

Scatter in Angular Chart component

Scatter

To render a scatter series, use series [type](#) as **Scatter** and inject **ScatterSeriesService** into the **@NgModule.providers**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChartModule } from '@syncfusion/ej2-angular-charts';
import { ScatterSeriesService, LegendService } from '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
import { ChartData } from './chartdata.service';
@Component({
    imports: [
        ChartModule
    ]
})

```

```

    ],
    providers: [ ScatterSeriesService, LegendService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='series1' type='Scatter' xName='x'
yName='y' name='Male' opacity=0.7 [marker]='marker'></e-series>
        <e-series [dataSource]='series2' type='Scatter' xName='x'
yName='y' name='Female' opacity=0.7 [marker]='marker'></e-series>
    </e-series-collection>
</ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    public series1?: Object;
    public series2?: Object;
    public marker?: Object;
    ngOnInit(): void {
      this.primaryXAxis = {
        title: 'Height (cm)',
        minimum: 120, maximum: 180,
        edgeLabelPlacement: 'Shift',
        labelFormat: '{value}cm'
      };
      this.primaryYAxis = {
        title: 'Weight (kg)',
        minimum: 60, maximum: 90,
        labelFormat: '{value}kg',
        rangePadding: 'None'
      };
      this.title = 'Height Vs Weight';
      this.marker = { width: 10, height: 10 };
      this.series1 = ChartData.prototype.getScatterData().series1;
      this.series2 = ChartData.prototype.getScatterData().series2;
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

The following properties can be used to customize the **scatter** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).

- [shape](#) - Specifies the shape of the scatter series.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ScatterSeriesService, LegendService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { ChartData } from './chartdata.service';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ ScatterSeriesService, LegendService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='series1' type='Scatter' xName='x'
yName='y' name='Male' opacity=0.9 [marker]='marker' fill='orange'></e-
series>
    <e-series [dataSource]='series2' type='Scatter' xName='x'
yName='y' name='Female' opacity=0.9 [marker]='marker' fill='red'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public series1?: Object;
  public series2?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.primaryXAxis = {
      title: 'Height (cm)',
      minimum: 120, maximum: 180,
      edgeLabelPlacement: 'Shift',
      labelFormat: '{value}cm'
    };
    this.primaryYAxis = {
      title: 'Weight (kg)',
      minimum: 60, maximum: 90,
      labelFormat: '{value}kg',
      rangePadding: 'None'
    };
    this.title = 'Height Vs Weight';
    this.marker = { width: 10, height: 10 };
    this.series1 = ChartData.prototype.getScatterData().series1;
    this.series2 = ChartData.prototype.getScatterData().series2;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Data label](#)
- [Tooltip](#)

Bubble in Angular Chart component

Bubble

To render a bubble series, use series [type](#) as **Bubble** and inject **BubbleSeriesService** into the **@NgModule.providers**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { BubbleSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { bubbleData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ BubbleSeriesService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart style='display:block;' id='chart-container'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
                [title]='title' >
                <e-series-collection>
                    <e-series [dataSource]='data' type='Bubble' xName='x'
yName='y' size='size' name='pound'> </e-series>
                </e-series-collection>
                </ejs-chart>`
})
export class AppComponent implements OnInit {
  public title?: string;
  public data?: Object[];
  primaryXAxis: any;
  primaryYAxis: any;
  ngOnInit(): void {
    this.data = bubbleData;
    this.title = 'GDP vs Literacy Rate';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Size mapping

size property can be used to map the size value specified in data source.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { BubbleSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { bubbleData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ BubbleSeriesService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart style='display:block;' id='chart-container'
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
    [title]='title' >
    <e-series-collection>
      <e-series [dataSource]='data' type='Bubble' xName='x'
        yName='y' size='size' name='pound'> </e-series>
    </e-series-collection>
  </ejs-chart> `
})
export class AppComponent implements OnInit {
  public title?: string;
  public data?: Object[];
  primaryXAxis: any;
  primaryYAxis: any;
  ngOnInit(): void {
    this.data = bubbleData;
    this.title = 'GDP vs Literacy Rate';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Series customization

The following properties can be used to customize the bubble series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { BubbleSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { bubbleData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ BubbleSeriesService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart style='display:block;' id='chart-container'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
                [title]='title' >
                <e-series-collection>
                    <e-series [dataSource]='data' type='Bubble' xName='x'
yName='y' size='size' name='pound' fill='blue' [border]='border'> </e-
series>
                </e-series-collection>
            </ejs-chart>`
})
export class AppComponent implements OnInit {
  public title?: string;
  public border: Object = {
    width: 1.5,
    color: 'black'
  };
  public data?: Object[];
  primaryXAxis: any;
  primaryYAxis: any;
  ngOnInit(): void {
    this.data = bubbleData;
    this.title = 'GDP vs Literacy Rate';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Data label](#)
- [Tooltip](#)

Polar in Angular Chart component

Polar

To render a polar series, use series [type](#) as **Polar** and inject **PolarSeriesService** into the **@NgModule.providers**.

Draw Types

Polar drawType property is used to change the series plotting type to line, column, area, range column, spline, scatter, stacking area and stacking column. The default value of drawType is **Line**.

Line

To render a line draw type, use series [drawType](#) as **Line** and inject **LineSeriesService** inject **LineSeriesService** into the **@NgModule.providers**. [isClosed](#) property specifies whether to join start and end point of a line series used in polar chart to form a closed path. Default value of isClosed is true.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService } from
'@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule, ButtonModule, ChartAllModule
  ],
  providers: [ AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart id='chartcontainer' [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis'
[title]='title' >
    <e-series-collection>
      <e-series [dataSource]='data' type='Polar' xName='x'
yName='y' drawType='Line'> </e-series>
    </e-series-collection>
  </ejs-chart> `
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public primaryYAxis?: Object;
  public data?: Object[];
  ngOnInit(): void {
    this.data = [{ x: 2005, y: 28 }, { x: 2006, y: 25 }, { x: 2007, y: 26
}, { x: 2008, y: 27 },
                { x: 2009, y: 32 }, { x: 2010, y: 35 }, { x: 2011, y:
30 }];
```

```

    this.primaryXAxis = {
      title: 'Year',
      minimum: 2004, maximum: 2012, interval: 1
    };
    this.primaryYAxis = {
      minimum: 20, maximum: 40, interval: 5,
      title: 'Efficiency',
      labelFormat: '{value}%'
    };
    this.title = 'Efficiency of oil-fired power production';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Spline

To render a spline line draw type, use series [drawType](#) as **Spline** and inject **SplineSeriesService** inject **SplineSeriesService** into the **@NgModule.providers**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService } from
 '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
import { polarCategory } from './datasource';
@Component({
  imports: [
    ChartModule, ButtonModule, ChartAllModule
  ],
  providers: [ AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart id='chartcontainer' [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis'
    [title]='title' >
      <e-series-collection>
        <e-series [dataSource]='data' type='Polar' xName='x'
yName='y' drawType='Spline' name='London'> </e-series>
      </e-series-collection>
    </ejs-chart> `
})

```

```

    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public title?: string;
        public primaryYAxis?: Object;
        public data?: Object[];
        ngOnInit(): void {
            this.data = polarCategory;
            this.primaryXAxis = {
                title: 'Month',
                valueType: 'Category'
            };
            this.primaryYAxis = {
                minimum: -5, maximum: 35, interval: 10,
                title: 'Temperature in Celsius',
                labelFormat: '{value}C'
            };
            this.title = 'Climate Graph-2012';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Area

To render a area line draw type, use series [drawType](#) as `Area` and inject `AreaSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService} from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { polarCategory } from './datasource';
@Component({
    imports: [
        ChartModule, ButtonModule, ChartAllModule
    ],
    providers: [ AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService],
    standalone: true,
    selector: 'app-container',

```

```

    template: `<ejs-chart id='chartcontainer' [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis'
    [title]='title' >
        <e-series-collection>
            <e-series [dataSource]='data' type='Polar' xName='x'
yName='y' drawType='Area' name='London'> </e-series>
        </e-series-collection>
    </ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public title?: string;
    public primaryYAxis?: Object;
    public data?: Object[];
    ngOnInit(): void {
      this.data = polarCategory;
      this.primaryXAxis = {
        title: 'Month',
        valueType: 'Category'
      };
      this.primaryYAxis = {
        minimum: -5, maximum: 35, interval: 10,
        title: 'Temperature in Celsius',
        labelFormat: '{value}C'
      };
      this.title = 'Climate Graph-2012';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Stacked Area

To render a stacked area draw type, use series `drawType` as `StackingArea` and inject `StackingAreaSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService } from
'@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { stackedData } from './datasource';
@Component({
  imports: [

```

```

    ChartModule, ButtonModule, ChartAllModule
  ],
  providers: [ AreaSeriesService, LineSeriesService, ExportService,
    ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
    RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
    CategoryService, RadarSeriesService, SplineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id='chartcontainer' [primaryXAxis]='primaryXAxis'
    [title]='title' >
    <e-series-collection>
      <e-series [dataSource]='data' type='Polar' xName='x'
yName='y' drawType='StackingArea' name='Organic'> </e-series>
      <e-series [dataSource]='data' type='Polar' xName='x'
yName='y1' drawType='StackingArea' name='Fair-trade'> </e-series>
      <e-series [dataSource]='data' type='Polar' xName='x'
yName='y2' drawType='StackingArea' name='veg-Alternatives'> </e-series>
    </e-series-collection>
    </ejs-chart>`
  })
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public data?: Object[];
  ngOnInit(): void {
    this.data = stackedData;
    this.primaryXAxis = {
      valueType: 'Category',
    };
    this.title = 'Trend in Sales of Ethical Produce';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Column

To render a column draw type, use series [drawType](#) as **Column** and inject **ColumnSeriesService** into the **@NgModule.providers**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { AreaSeriesService, LineSeriesService, ExportService,
  ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
  RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
  CategoryService, RadarSeriesService, SplineSeriesService } from
  '@syncfusion/ej2-angular-charts'

```

```

import { Component, OnInit } from '@angular/core';
import { columnData } from './datasource';
@Component({
  imports: [
    ChartModule, ButtonModule, ChartAllModule
  ],
  providers: [ AreaSeriesService, LineSeriesService, ExportService,
    ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
    RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
    CategoryService, RadarSeriesService, SplineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart id='chartcontainer' [primaryXAxis]='primaryXAxis'
    [title]='title' >
    <e-series-collection>
      <e-series [dataSource]='data' type='Polar' xName='country'
yName='gold' drawType='Column' name='Gold'> </e-series>
    </e-series-collection>
  </ejs-chart> `
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public data?: Object[];
  ngOnInit(): void {
    this.data = columnData;
    this.primaryXAxis = {
      valueType: 'Category',
      title: 'Countries'
    };
    this.title = 'Olympic Medals';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Stacked Column

To render a stacked column draw type, use series [drawType](#) as `StackingColumn` and inject `StackingColumnSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { AreaSeriesService, LineSeriesService, ExportService,
  ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
  RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,

```

```

CategoryService, RadarSeriesService, SplineSeriesService} from
'@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { data } from './datasource';
@Component({
  imports: [
    ChartModule, ButtonModule, ChartAllModule
  ],
  providers: [ AreaSeriesService, LineSeriesService, ExportService,
    ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
    RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
    CategoryService, RadarSeriesService, SplineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart id='chartcontainer' [primaryXAxis]='primaryXAxis'
    [title]='title' >
    <e-series-collection>
      <e-series [dataSource]='data' type='Polar' xName='x'
yName='y' drawType='StackingColumn' name='UK'> </e-series>
      <e-series [dataSource]='data' type='Polar' xName='x'
yName='y1' drawType='StackingColumn' name='Germany'> </e-series>
      <e-series [dataSource]='data' type='Polar' xName='x'
yName='y2' drawType='StackingColumn' name='France'> </e-series>
      <e-series [dataSource]='data' type='Polar' xName='x'
yName='y2' drawType='StackingColumn' name='Italy'> </e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public data?: Object[];
  ngOnInit(): void {
    this.data = data;
    this.primaryXAxis = {
      title: 'Years',
      valueType: 'Category'
    };
    this.title = 'Mobile Game Market by Country';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Range Column

To render a range column draw type, use series `drawType` as `RangeColumn` and inject `RangeColumnSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS


```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { rangeData } from './datasource';
@Component({
  imports: [
    ChartModule, ButtonModule, ChartAllModule
  ],
  providers: [ AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart id='chartcontainer' [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis'
    [title]='title' >
      <e-series-collection>
        <e-series [dataSource]='data' type='Polar' xName='x'
high='high' low='low' drawType='RangeColumn' name='Gold'> </e-series>
      </e-series-collection>
    </ejs-chart> `
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public primaryYAxis?: Object;
  public data?: Object[];
  ngOnInit(): void {
    this.data = rangeData;
    this.primaryXAxis = {
      valueType: 'Category',
      title: 'Months'
    };
    this.primaryYAxis = {
      title: 'Temperature(Celsius)',
    };
    this.title = 'Maximum and Minimum Temperature';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Scatter

To render a scatter draw type, use series [drawType](#) as `Scatter` and inject `ScatterSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { polarCategory } from './datasource';
@Component({
  imports: [
    ChartModule, ButtonModule, ChartAllModule
  ],
  providers: [ AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart id='chartcontainer' [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis'
    [title]='title' >
      <e-series-collection>
        <e-series [dataSource]='data' type='Polar' xName='x'
yName='y' drawType='Scatter' name='London'> </e-series>
      </e-series-collection>
    </ejs-chart> `
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public primaryYAxis?: Object;
  public data?: Object[];
  ngOnInit(): void {
    this.data = polarCategory;
    this.primaryXAxis = {
      title: 'Month',
      valueType: 'Category'
    };
    this.primaryYAxis = {
      minimum: -5, maximum: 35, interval: 10,
      title: 'Temperature in Celsius',
      labelFormat: '{value}C'
    };
    this.title = 'Climate Graph-2012';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

*Series Customization**Start angle*

You can customize the start angle of the polar series using [startAngle](#) property. By default, **startAngle** is 0 degree.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { radarData } from './datasource';
@Component({
  imports: [
    ChartModule, ButtonModule, ChartAllModule
  ],
  providers: [ AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart id='chartcontainer' [primaryXAxis]='primaryXAxis'
    [title]='title' >
    <e-series-collection>
      <e-series [dataSource]='data' type='Polar' xName='x'
yName='y' drawType='Line'> </e-series>
    </e-series-collection>
  </ejs-chart> `
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public primaryYAxis?: Object;
  public data?: Object[];
  ngOnInit(): void {
    this.data = radarData;
    this.primaryXAxis = {
      title: 'Year', startAngle: 90,
      minimum: 2004, maximum: 2012, interval: 1
    };
    this.title = 'Efficiency of oil-fired power production';
  }
}
```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Radius

You can customize the radius of the polar series and polar series using [coefficient](#) property. By default, [coefficient](#) is 100.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService} from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { radarData } from './datasource';
@Component({
  imports: [
    ChartModule, ButtonModule, ChartAllModule
  ],
  providers: [ AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart id='chartcontainer' [primaryXAxis]='primaryXAxis'
    [title]='title' >
    <e-series-collection>
      <e-series [dataSource]='data' type='Polar' xName='x'
yName='y' drawType='Line'> </e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public data?: Object[];
  ngOnInit(): void {
    this.data = radarData;
    this.primaryXAxis = {
      title: 'Year', coefficient: 50,
      minimum: 2004, maximum: 2012, interval: 1
    };
    this.title = 'Efficiency of oil-fired power production';
```

```
}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Data label](#)
- [Tooltip](#)

Radar in Angular Chart component

Radar

To render a radar series, use series [type](#) as **Radar** and inject **RadarSeriesService** into the **@NgModule.providers**.

To render a line draw type, use series [drawType](#) as **Line** and inject **LineSeriesService** inject **LineSeriesService** into the **@NgModule.providers**. [isClosed](#) property specifies whether to join start and end point of a line series used in polar chart to form a closed path. Default value of [isClosed](#) is true.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { radarData } from './datasource';
@Component({
  imports: [
    ChartModule, ButtonModule, ChartAllModule
  ],
  providers: [ AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id='chartcontainer' [title]='title' >
    <e-series-collection>
      <e-series [dataSource]='data' type='Radar' xName='x'
yName='y' drawType='Line'> </e-series>
    </e-series-collection>
  </ejs-chart>`
```

```

    })
    export class AppComponent implements OnInit {
        public title?: string;
        public data?: Object[];
        ngOnInit(): void {
            this.data = radarData;
            this.title = 'Efficiency of oil-fired power production';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

Start Angle

You can customize the start angle of the polar series using [startAngle](#) property. By default, `startAngle` is 0 degree.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { radarData } from './datasource';
@Component({
    imports: [
        ChartModule, ButtonModule, ChartAllModule
    ],
    providers: [ AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService],
    standalone: true,
    selector: 'app-container',
    template: ` <ejs-chart id='chartcontainer' [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis'
                [title]='title' >
                <e-series-collection>
                    <e-series [dataSource]='data' type='Radar' xName='x'
yName='y' drawType='Line'> </e-series>
                </e-series-collection>
            </ejs-chart> `
})
export class AppComponent implements OnInit {

```

```

public primaryXAxis?: Object;
public title?: string;
public primaryYAxis?: Object;
public data?: Object[];
ngOnInit(): void {
    this.data = radarData;
    this.primaryXAxis = {
        title: 'Year', startAngle: 90,
        minimum: 2004, maximum: 2012, interval: 1
    };
    this.title = 'Efficiency of oil-fired power production';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Radius

You can customize the radius of the polar series and radar series using [coefficient](#) property. By default, **coefficient** is 100.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService} from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { radarData } from './datasource';
@Component({
  imports: [
    ChartModule, ButtonModule, ChartAllModule
  ],
  providers: [ AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart id='chartcontainer' [primaryXAxis]='primaryXAxis'
    [title]='title' >
    <e-series-collection>
      <e-series [dataSource]='data' type='Radar' xName='x'
yName='y' drawType='Line'> </e-series>
    </e-series-collection>
  </ejs-chart>`
})

```

```

    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public title?: string;
        public data?: Object[];
        ngOnInit(): void {
            this.data = radarData;
            this.primaryXAxis = {
                title: 'Year', coefficient: 50,
                minimum: 2004, maximum: 2012, interval: 1
            };
            this.title = 'Efficiency of oil-fired power production';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Data label](#)
- [Tooltip](#)

Hilo in Angular Chart component

Hilo

Hilo Series illustrates the price movements in stock using the high and low values.

To render a Hilo series, use series [type](#) as **Hilo** and inject **HiloSeriesService** into the **@NgModule.providers**.

Hilo series requires 3 fields (x, high and low) to show the high and low price in the stock.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, HiloSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
    imports: [
        ChartModule
    ],
    providers: [CategoryService, HiloSeriesService],
    standalone: true,
    selector: 'app-container',
    template: ` <ejs-chart style='display:block;' id='chart-container'
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
    [title]='title' >

```



```

        <e-series-collection>
            <e-series [dataSource]='data' type='Hilo' xName='x'
high='high' low='low' name='India'> </e-series>
        </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public title?: string;
    public primaryYAxis?: Object;
    public data?: Object[];
    ngOnInit(): void {
        this.data = [
            { x: 'Jan', low: 87, high: 200 }, { x: 'Feb', low: 45, high: 135
},
            { x: 'Mar', low: 19, high: 85 }, { x: 'Apr', low: 31, high: 108
},
            { x: 'May', low: 27, high: 80 }, { x: 'June', low: 84, high: 130
},
            { x: 'July', low: 77, high: 150 }, { x: 'Aug', low: 54, high:
125 },
            { x: 'Sep', low: 60, high: 155 }, { x: 'Oct', low: 60, high: 180
},
            { x: 'Nov', low: 88, high: 180 }, { x: 'Dec', low: 84, high: 230
}
        ];
        this.primaryXAxis = {
            valueType: 'Category',
            title: 'Months'
        };
        this.primaryYAxis = {
            labelFormat: '{value}mm',
            edgeLabelPlacement: 'Shift',
            title: 'Rainfall',
        };
        this.title = 'Maximum and Minimum Rainfall';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

The following properties can be used to customize the **hilo** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService,HiloSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [CategoryService,HiloSeriesService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart style='display:block;' id='chart-container'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
                [title]='title' >
                <e-series-collection>
                    <e-series [dataSource]='data' type='Hilo' xName='x'
high='high' low='low' name='India' fill='blue'> </e-series>
                </e-series-collection>
            </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public primaryYAxis?: Object;
  public data?: Object[];
  ngOnInit(): void {
    this.data = [
      { x: 'Jan', low: 87, high: 200 }, { x: 'Feb', low: 45, high: 135
    },
      { x: 'Mar', low: 19, high: 85 }, { x: 'Apr', low: 31, high: 108
    },
      { x: 'May', low: 27, high: 80 }, { x: 'June', low: 84, high: 130
    },
      { x: 'July', low: 77, high: 150 }, { x: 'Aug', low: 54, high:
125 },
      { x: 'Sep', low: 60, high: 155 }, { x: 'Oct', low: 60, high: 180
    },
      { x: 'Nov', low: 88, high: 180 }, { x: 'Dec', low: 84, high: 230
    }
    ];
    this.primaryXAxis = {
      valueType: 'Category',
      title: 'Months'
    };
    this.primaryYAxis = {
      labelFormat: '{value}mm',
      edgeLabelPlacement: 'Shift',
      title: 'Rainfall',
    };
    this.title = 'Maximum and Minimum Rainfall';
  }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Data label](#)
- [Tooltip](#)

High Low Open Close in Angular Chart component

High Low Open Close

HiloOpenClose series is used to represent the low, high, open and closing values over time.

To render a HiloOpenClose series, use series [type](#) as HiloOpenClose and inject HiloOpenCloseSeriesService into the @NgModule.providers.

HiloOpenClose series requires 5 fields (x, high, low, open and close) to show the high, low, open and close price values in the stock.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, HiloOpenCloseSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [CategoryService, HiloOpenCloseSeriesService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart style='display:block;' id='chart-container'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[title]='title' >
    <e-series-collection>
      <e-series [dataSource]='data' type='HiloOpenClose'
xName='x' high='high' low='low' open='open' close='close' name='SHIRPUR-G'>
    </e-series>
    </e-series-collection>
  </ejs-chart> `
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public primaryYAxis?: Object;
  public data?: Object[];
  ngOnInit(): void {
    this.data = [
      { x: 'Jan', open: 120, high: 160, low: 100, close: 140 },
      { x: 'Feb', open: 150, high: 190, low: 130, close: 170 },
      { x: 'Mar', open: 130, high: 170, low: 110, close: 150 },
```

```

        { x: 'Apr', open: 160, high: 180, low: 120, close: 140 },
        { x: 'May', open: 150, high: 170, low: 110, close: 130 }
    ];
    this.primaryXAxis = {
        title: 'Date',
        valueType: 'Category',
    };
    this.primaryYAxis = {
        title: 'Price in Dollar', minimum: 100, maximum: 200, interval:
20,
    };
    this.title = 'Financial Analysis';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

In HiloOpenClose series, [bullFillColor](#) is used to fill the segment when the open value is greater than the close value and [bearFillColor](#) is used to fill the segment when the open value is less than the close value.

By default, bullFillColor is set as red and bearFillColor is set as green.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChartModule } from '@syncfusion/ej2-angular-charts';
import { CategoryService, HiloOpenCloseSeriesService } from '@syncfusion/ej2-
angular-charts';
import { Component, OnInit } from '@angular/core';
import { openData } from './datasource';
@Component({
    imports: [
        ChartModule
    ],
    providers: [CategoryService, HiloOpenCloseSeriesService],
    standalone: true,
    selector: 'app-container',
    template: ` <ejs-chart style='display:block;' id='chart-container'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[title]='title' >
<e-series-collection>
    <e-series [dataSource]='data' type='HiloOpenClose'
xName='x' high='high' low='low' open='open' close='close' name='SHIRPUR-G'
bearFillColor= '#e56590' bullFillColor= '#f8b883'> </e-series>
    </e-series-collection>
</ejs-chart> `
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;

```

```

public title?: string;
public primaryYAxis?: Object;
public data?: Object[];
ngOnInit(): void {
    this.data = openData;
    this.primaryXAxis = {
        title: 'Date',
        valueType: 'Category',
    };
    this.primaryYAxis = {
        title: 'Price in Dollar', minimum: 100, maximum: 200, interval:
20,
    };
    this.title = 'Financial Analysis';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Data label](#)
- [Tooltip](#)

Candle in Angular Chart component

Candle

Candle series are similar to Hilo Open Close series, are used to represent the low, high, open and closing price over time. To render a candle series, use series [type](#) as `Candle` and inject `CandleSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, CandleSeriesService } from '@syncfusion/ej2-
angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
    imports: [
        ChartModule
    ],
    providers: [CategoryService, CandleSeriesService],
    standalone: true,
    selector: 'app-container',
    template: ` <ejs-chart style='display:block;' id='chart-container'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
>
                <e-series-collection>

```

```

        <e-series [dataSource]='data' type='Candle' xName='x'
high='high' low='low' open='open' close='close' name='SHIRPUR-G'> </e-
series>
        </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public title?: string;
    public primaryYAxis?: Object;
    public data?: Object[];
    ngOnInit(): void {
        this.data = [
            { x: 'Jan', open: 120, high: 160, low: 100, close: 140 },
            { x: 'Feb', open: 150, high: 190, low: 130, close: 170 },
            { x: 'Mar', open: 130, high: 170, low: 110, close: 150 },
            { x: 'Apr', open: 160, high: 180, low: 120, close: 140 },
            { x: 'May', open: 150, high: 170, low: 110, close: 130 }
        ];
        this.primaryXAxis = {
            title: 'Date',
            valueType: 'Category',
        };
        this.primaryYAxis = {
            title: 'Price in Dollar', minimum: 100, maximum: 200, interval:
20,
        };
        this.title = 'Financial Analysis';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Hollow Candles

Candle charts allow to visually compare the current price with previous price by customizing the appearance.

Candles are filled/left as hollow based on the following criteria.

<!-- markdownlint-disable MD033 -->

States	Description
Filled	candle sticks are filled when the close value is lesser than the open value
Unfilled	candle sticks are unfilled when the close value is greater than the open value

The color of the candle will be defined by comparing with previous values.

Bear color will be applied when the current closing value is greater than the previous closing value.

Bull color will be applied when the current closing value is less than the previous closing value.

By default, bullFillColor is set as red and bearFillColor is set as green.

Solid Candles

[enableSolidCandles](#) is used to enable/disable the solid candles. By default is set to be false. The fill color of the candle will be defined by its opening and closing values.

[bearFillColor](#) will be applied when the opening value is less than the closing value.

[bullFillColor](#) will be applied when the opening value is greater than closing value.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, CandleSeriesService } from '@syncfusion/ej2-
angular-charts'
import { Component, OnInit } from '@angular/core';
import { candleData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [CategoryService, CandleSeriesService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart style='display:block;' id='chart-container'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[title]='title'>
    <e-series-collection>
      <e-series [dataSource]='data' type='Candle' xName='x'
high='high' low='low' open='open' close='close' name='SHIRPUR-G'
bearFillColor= '#e56590' bullFillColor= '#f8b883'
[enableSolidCandles]='enableSolidCandles'> </e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public primaryYAxis?: Object;
  public data?: Object[];
  public enableSolidCandles: Object = { enable: true };
  ngOnInit(): void {
    this.data = candleData;
    this.primaryXAxis = {
      title: 'Date',
      valueType: 'Category',
    };
    this.primaryYAxis = {
      title: 'Price in Dollar', minimum: 100, maximum: 200, interval:
20,
    };
    this.title = 'Financial Analysis';
```

```
}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Data label](#)
- [Tooltip](#)

Box and Whisker in Angular Chart component

Box and Whisker

To render a box and whisker chart, use series [type](#) as `BoxAndWhisker` and inject

`BoxAndWhiskerSeriesService` into the `@NgModule.providers`. The field `y` requires n number of data or it should contains minimum of five values to plot a segment.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BoxAndWhiskerSeriesService, DataLabelService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { data } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [CategoryService, BoxAndWhiskerSeriesService, DataLabelService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id='chart-container'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
    [title]='title' >
    <e-series-collection>
      <e-series [dataSource]='data' type='BoxAndWhisker' xName='x'
yName='y' [marker]='marker'> </e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public data?: Object[];
  public marker?: Object;
  public tooltip?: Object;
```



```

primaryYAxis: any;
ngOnInit(): void {
  this.data = data;
  this.primaryXAxis = {
    valueType: 'Category',
  };
  this.title = 'Company Revenue and Profit';
  this.marker = { visible: true }
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Box plot

You can change the rendering mode of the Box and Whisker series using the `boxPlotMode` property.

The default `boxPlotMode` is `exclusive`. The other `boxPlotMode` available are `inclusive` and `normal`.

To render a box and whisker chart, use series `type` as `BoxAndWhisker` and inject `BoxAndWhiskerSeriesService` into the `@NgModule.providers`. The field `y` requires n number of data or it should contains minimum of five values to plot a segment.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BoxAndWhiskerSeriesService, DataLabelService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { data } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [CategoryService, BoxAndWhiskerSeriesService, DataLabelService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart id='chart-container'
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
    [title]='title' >
      <e-series-collection>
        <e-series [dataSource]='data' type='BoxAndWhisker' xName='x'
          yName='y' [marker]='marker' boxPlotMode='Normal'> </e-series>
      </e-series-collection>
    </ejs-chart> `
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;

```

```

public title?: string;
public primaryYAxis?: Object;
public data?: Object[];
public marker?: Object;
public tooltip?: Object;
ngOnInit(): void {
    this.data = data;
    this.primaryXAxis = {
        valueType: 'Category',
    };
    this.title = 'Company Revenue and Profit';
    this.marker = { visible: true };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Show mean

In Box and Whisker series **showMean** property is used to show the box and whisker average value. The default value of **showMean** is false.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BoxAndWhiskerSeriesService, DataLabelService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { data } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [CategoryService, BoxAndWhiskerSeriesService, DataLabelService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart id='chartcontainer' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title' >
    <e-series-collection>
      <e-series [dataSource]='data' type='BoxAndWhisker' xName='x' yName='y' [marker]='marker' [showMean]='showMean'> </e-series>
    </e-series-collection>
  </ejs-chart> `
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public primaryYAxis?: Object;

```

```

public data?: Object[];
public marker?: Object;
public tooltip?: Object;
public showMean: boolean = false;
ngOnInit(): void {
    this.data = data;
    this.primaryXAxis = {
        valueType: 'Category',
    };
    this.marker = {
        visible: true
    };
    this.title = 'Company Revenue and Profit';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Data label](#)
- [Tooltip](#)

Waterfall in Angular Chart component

Waterfall Chart

Waterfall chart helps to understand the cumulative effect of the sequentially introduced positive and negative values. To render a Waterfall series, use series [type](#) as

Waterfall and inject **WaterfallSeriesService** into the **@NgModule.providers**.

[intermediateSumIndexes](#) property of waterfall is used to represent the in between the sum values and [sumIndexes](#) is used to represent the cumulative sum values.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, WaterfallSeriesService, DataLabelService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [CategoryService, WaterfallSeriesService, DataLabelService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart style='display:block;' id='chart-container'
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'

```

```

        [title]='title' >
        <e-series-collection>
            <e-series [dataSource]='data' type='Waterfall' xName='x'
yName='y' name='USA' [columnWidth]='columnWidth'
            [connector]='connector'
[intermediateSumIndexes]='intermediate' [sumIndexes]='sum'
[marker]='marker'> </e-series>
            </e-series-collection>
        </ejs-chart>`
    ))
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public title?: string;
    public primaryYAxis?: Object;
    public data?: Object[];
    public marker?: Object;
    public connector?: Object;
    public sum: number[] = [8];
    public intermediate: number[] = [4, 7];
    public columnWidth: number = 0.6;
    ngOnInit(): void {
        this.data = [
            { x: 'Income', y: 4711 }, { x: 'Sales', y: -1015 },
            { x: 'Development', y: -688 },
            { x: 'Revenue', y: 1030 }, {x: 'Balance'},
            { x: 'Administrative', y: -780 },
            { x: 'Expense', y: -361 }, { x: 'Tax', y: -695 },
            { x: 'Net Profit' }
        ];
        this.primaryXAxis = {
            majorGridLines: {width: 0},
            valueType: 'Category',
        };
        this.primaryYAxis = {
            labelFormat: '${value}M',
            minimum: 0, maximum: 5500, interval: 500,
            majorGridLines: {width: 0},
            lineStyle: { width: 0},
            majorTickLines: { width: 0}
        };
        this.marker = {
            dataLabel: { visible: true, position: 'Outer' }
        };
        this.connector = { color: '#5F6A6A', width: 1.5 };
        this.title = 'Company Revenue and Profit';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series customization

The negative changes of waterfall charts is represented by using [negativeFillColor](#) and the summary changes are represented by using [summaryFillColor](#) properties.

By default, the negativeFillColor as '#E94649' and the summaryFillColor as '#4E81BC'.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, WaterfallSeriesService, DataLabelService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { waterfallData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [CategoryService, WaterfallSeriesService, DataLabelService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart style='display:block;' id='chart-container'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[title]='title' >
<e-series-collection>
<e-series [dataSource]='data' type='Waterfall' xName='x'
yName='y' name='USA' [columnWidth]='columnWidth' summaryFillColor='#e56590'
negativeFillColor='#f8b883'
[connector]='connector'
[intermediateSumIndexes]='intermediate' [sumIndexes]='sum'
[marker]='marker'> </e-series>
</e-series-collection>
</ejs-chart> `
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public primaryYAxis?: Object;
  public data?: Object[];
  public marker?: Object;
  public connector?: Object;
  public sum: number[] = [8];
  public intermediate: number[] = [4, 7];
  public columnWidth: number = 0.6;
  ngOnInit(): void {
    this.data = waterfallData;
    this.primaryXAxis = {
      valueType: 'Category',
    };
    this.primaryYAxis = {
      labelFormat: '${value}M',
      minimum: 0, maximum: 5500, interval: 500,
    };
    this.marker = {
      dataLabel: { visible: true, position: 'Outer' }
    };
  }
}
```

```

        this.connector = { color: '#5F6A6A', width: 1.5 };
        this.title = 'Company Revenue and Profit';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Data label](#)
- [Tooltip](#)

Histogram in Angular Chart component

Histogram

Histogram type charts can provide a visual display of large amounts of data that are difficult to understand in a tabular or spreadsheet form. To render a histogram chart, use series [type](#) as `Histogram` and inject `HistogramSeriesService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { HistogramSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { ILoadedEventArgs } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    ChartModule
  ],
  providers: [HistogramSeriesService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-chart style='display:block;' align='center' id='chart-
container' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
(load)='load($event)'>
    <e-series-collection>
      <e-series [dataSource]='data' type='Histogram' yName='y'
name='Score' width=2 [binInterval]='binInterval'
showNormalDistribution='showNormalDistribution'
[columnWidth]='columnWidth'> </e-series>
    </e-series-collection>
  </ejs-chart> `
})
export class AppComponent implements OnInit {
  public data: Object[] = [];
  public primaryXAxis: Object = {
    minimum: 0, maximum: 100
  };
};

```

```

    public primaryYAxis: Object = {
        minimum: 0, maximum: 50, interval: 10,
    };
    public load(args: ILoadedEventArgs): void {
        let points: number[] = [5.250, 7.750, 0, 8.275, 9.750, 7.750, 8.275,
6.250, 5.750,
        5.250, 23.000, 26.500, 27.750, 25.025, 26.500, 26.500, 28.025,
29.250, 26.750, 27.250,
        26.250, 25.250, 34.500, 25.625, 25.500, 26.625, 36.275, 36.250,
26.875, 40.000, 43.000,
        46.500, 47.750, 45.025, 56.500, 56.500, 58.025, 59.250, 56.750,
57.250,
        46.250, 55.250, 44.500, 45.525, 55.500, 46.625, 46.275, 56.250,
46.875, 43.000,
        46.250, 55.250, 44.500, 45.425, 55.500, 56.625, 46.275, 56.250,
46.875, 43.000,
        46.250, 55.250, 44.500, 45.425, 55.500, 46.625, 56.275, 46.250,
56.875, 41.000, 63.000,
        66.500, 67.750, 65.025, 66.500, 76.500, 78.025, 79.250, 76.750,
77.250,
        66.250, 75.250, 74.500, 65.625, 75.500, 76.625, 76.275, 66.250,
66.875, 80.000, 85.250,
        87.750, 89.000, 88.275, 89.750, 97.750, 98.275, 96.250, 95.750,
95.250
        ];
        points.map((value: number) => {
            this.data.push({
                y: value
            });
        });
    };
    public binInterval: number = 20;
    public columnWidth: number = 0.99;
    public showNormalDistribution: boolean = true;
    ngOnInit(): void {
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Data label](#)
- [Tooltip](#)

Error Bar in Angular Chart component

Error bar

Error bars are graphical representations of the variability of data and used on graphs to indicate the error or uncertainty in a reported measurement. To render the error bar for the series, set [visible](#) as `true` in error bar object and inject `ErrorBar` module into the `@NgModule.providers`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ColumnSeriesService, LineSeriesService, ErrorBarService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ColumnSeriesService, LineSeriesService, ErrorBarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='India' width=2 [marker]='marker' [errorBar]='errorBar'></e-
series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  public errorBar?: Object;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = [
      { x: 2006, y: 7.8 }, { x: 2007, y: 7.2 },
      { x: 2008, y: 6.8 }, { x: 2009, y: 10.7 },
      { x: 2010, y: 10.8 }, { x: 2011, y: 9.8 }
    ];
    this.primaryXAxis = {
      minimum: 2005, maximum: 2012, interval: 1,
      title: 'Year'
    };
    this.marker = { visible: true };
    this.errorBar = { visible: true };
    this.title = 'Unemployment rate (%)';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
```



```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Error bar type

To change the error bar rendering type using [type](#) option of error bar. To change the error bar line length you can use [verticalError](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ColumnSeriesService, LineSeriesService, ErrorBarService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { errorData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ColumnSeriesService, LineSeriesService, ErrorBarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='India' width=2 [marker]='marker' [errorBar]='errorBar'></e-
series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  public errorBar?: Object;
  primaryXAxis: any;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = errorData;
    this.marker = { visible: true };
    this.errorBar = { visible: true, type: 'Percentage',
verticalError:4 };
    this.title = 'Unemployment rate (%)';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customizing error bar type

To customize the error bar type, set error bar [type](#) as **Custom** and then change the horizontal/vertical positive and negative error of error bar.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ColumnSeriesService, LineSeriesService, ErrorBarService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { errorData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ColumnSeriesService, LineSeriesService, ErrorBarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='India' width=2 [marker]='marker' [errorBar]='errorBar'></e-
series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  public errorBar?: Object;
  primaryXAxis: any;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = errorData;
    this.marker = { visible: true };
    this.errorBar = {
      visible: true,
      type: 'Custom',
      mode: 'Both',
      verticalPostiveError: 3,
      horizontalPositiveError: 2,
      verticalNegativeError: 3,
      horizontalNegativeError: 2
    } as Object;
    this.title = 'Unemployment rate (%)';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Error bar mode

Error bar mode is used to define whether the error bar line has to be drawn horizontally, vertically or in both side. To change the error bar mode use [mode](#) option.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ColumnSeriesService, LineSeriesService, ErrorBarService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { errorData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ColumnSeriesService, LineSeriesService, ErrorBarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='India' width=2 [marker]='marker' [errorBar]='errorBar'></e-
series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  public errorBar?: Object;
  primaryXAxis: any;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = errorData;
    this.marker = { visible: true };
    this.errorBar = { visible: true, mode: 'Horizontal' };
    this.title = 'Unemployment rate (%)';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Error bar direction

To change the error bar direction to plus, minus or both side using [direction](#) option.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ColumnSeriesService, LineSeriesService, ErrorBarService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { errorData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ColumnSeriesService, LineSeriesService, ErrorBarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='India' width=2 [marker]='marker' [errorBar]='errorBar'></e-
series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  public errorBar?: Object;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = errorData;
    this.marker = { visible: true };
    this.errorBar = { visible: true, mode: 'Vertical', direction: 'Minus'
  };
    this.title = 'Unemployment rate (%)';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customizing error bar cap

To customize the error bar cap length, width and fill color, you can use [errorBarCap](#) option.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ColumnSeriesService, LineSeriesService, ErrorBarService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { errorData } from '../datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ColumnSeriesService, LineSeriesService, ErrorBarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='India' width=2 [marker]='marker' [errorBar]='errorBar'></e-
series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  public errorBar?: Object;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = errorData;
    this.primaryXAxis = {
      minimum: 2005, maximum: 2012, interval: 1,
      title: 'Year'
    };
    this.marker = { visible: true };
    this.errorBar = { visible: true, errorBarCap:{ length:10, width:10,
color:'#0000ff'
} };
    this.title = 'Unemployment rate (%)';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customizing error bar color

To customize the error bar color for individual errors, use the [errorBarColorMapping](#) property. You can also customize the vertical error, horizontal error, horizontal negative and positive error and vertical negative and positive error for an individual point using [verticalError](#), [horizontalError](#), [horizontalNegativeError](#), [horizontalPositiveError](#), [verticalNegativeError](#) and [verticalPositiveError](#) properties.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ColumnSeriesService, LineSeriesService, ErrorBarService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ColumnSeriesService, LineSeriesService, ErrorBarService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-chart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
      <e-series-collection>
        <e-series [dataSource]='chartData' type='Line' xName='x'
        yName='y' name='India' width=2 [marker]='marker' [errorBar]='errorBar'></e-
        series>
      </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData: Object[] = [
    { x: 2006, y: 7.8, color: 'red', error: 1.5 },
    { x: 2007, y: 7.2, color: 'green', error: 2.5 },
    { x: 2008, y: 6.8, color: 'blue', error: 3.5 },
    { x: 2009, y: 5.7, color: 'yellow', error: 1.5 },
    { x: 2010, y: 10.8, color: 'grey', error: 0.5 },
    { x: 2011, y: 9.8, color: 'brown', error: 1 },
  ];
  public title?: string;
  public marker?: Object;
  public errorBar?: Object;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.primaryXAxis = {
      minimum: 2005, maximum: 2012, interval: 1,
      title: 'Year'
    };
    this.marker = { visible: true };
    this.errorBar = { visible: true, errorBarColorMapping: 'color',
    verticalError: 'error' };
    this.title = 'Unemployment rate (%)';
  }
}
```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Data label](#)
- [Tooltip](#)

Vertical Chart in Angular Chart component

Vertical Chart

In EJ2 chart, you can draw a chart in vertical manner by changing orientation of the axis. All series types support this feature.

You can use `isTransposed` property in chart to render a chart in vertical manner.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, StepLineSeriesService,
    SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
    ParetoSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { splineData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService, StepLineSeriesService,
    SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
    ParetoSeriesService, ColumnSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
isTransposed='true'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Spline' xName='x'
yName='y' name='London' width=2 [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
```

```

public chartData?: Object[];
public title?: string;
public primaryYAxis?: Object;
public marker?: Object;
ngOnInit(): void {
    this.chartData = splineData;
    this.primaryXAxis = {
        title: 'Month',
        valueType: 'Category'
    };
    this.marker = { visible: true, width: 10, height: 10 };
    this.primaryYAxis = {
        minimum: -5, maximum: 35, interval: 5,
        title: 'Temperature in Celsius',
        labelFormat: '{value}C'
    };
    this.title = 'Climate Graph-2012';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Data label](#)
- [Tooltip](#)

Pareto in Angular Chart component

Pareto

Pareto charts are used to find the cumulative values of data in different categories. It is a combination of Column and Line series.

The initial values are represented by column chart and the cumulative values are represented by Line chart.

To render a pareto chart, use series [type](#) as **Pareto** and inject **ParetoSeries** **ColumnSeries** and **LineSeries** module.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, StepLineSeriesService,
    SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
    ParetoSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';

```



```

import { splineData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService, StepLineSeriesService,
    SplineSeriesService, StackingLineSeriesService, DateTimeService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
    ParetoSeriesService, ColumnSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[tooltip]='tooltip'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Pareto' xName='x'
yName='y' name='Defect' width=2 [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
  public tooltip?: Object;
  ngOnInit(): void {
    this.chartData = [
      { x: 'Traffic', y: 56 }, { x: 'Child Care', y: 44.8 },
      { x: 'Transport', y: 27.2 }, { x: 'Weather', y: 19.6 },
      { x: 'Emergency', y: 6.6 }
    ];
    this.primaryXAxis = {
      title: 'Defects',
      valueType: 'Category',
    };
    this.marker = { visible: true, width: 10, height: 10 };
    this.primaryYAxis = {
      title: 'Frequency',
      minimum: 0,
      maximum: 150,
      interval: 30,
    };
    this.tooltip = { enable: true, shared: true };
    this.title = 'Climate Graph-2012';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Pareto customization

The pareto line series can be customized by using the [marker](#), [width](#), [dashArray](#), and [fill](#) properties in the [paretoOptions](#). The secondary axis for the pareto series can be shown or hidden using the [showAxis](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, StepLineSeriesService,
SplineSeriesService, StackingLineSeriesService, DateTimeService,
SplineAreaSeriesService, MultiColoredLineSeriesService,
ParetoSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { Browser } from '@syncfusion/ej2-base';
@Component({
imports: [
ChartModule
],
providers: [ CategoryService, LineSeriesService, StepLineSeriesService,
SplineSeriesService, StackingLineSeriesService, DateTimeService,
SplineAreaSeriesService, MultiColoredLineSeriesService,
ParetoSeriesService, ColumnSeriesService],
standalone: true,
selector: 'app-container',
template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[tooltip]='tooltip'>
<e-series-collection>
<e-series [dataSource]='chartData' type='Pareto' xName='x'
yName='y' name='Defect' width=2 opacity="0.75" [cornerRadius]="cornerRadius"
columnWidth="0.4" [paretoOptions]='paretoOptions'></e-series>
</e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
public primaryXAxis?: Object;
public chartData?: Object[];
public title?: string;
public primaryYAxis?: Object;
public tooltip?: Object;
public cornerRadius: Object = {
topLeft: Browser.isDevice ? 4 : 6, topRight: Browser.isDevice ? 4 :
6
};
public paretoOptions: Object = {
marker: {
visible: true,
isFilled: true,
width: 7,
height: 7
},
dashArray: '3,2',
width: 2,
fill: '#F7523F'
}
```

```

    };
    ngOnInit(): void {
      this.chartData = [
        { x: 'Button Defect', y: 23 }, { x: 'Pocket Defect', y: 16 },
        { x: 'Collar Defect', y: 10 }, { x: 'Cuff Defect', y: 7 },
        { x: 'Sleeve Defect', y: 6 }, { x: 'Other Defect', y: 2 }
      ];
      this.primaryXAxis = {
        interval: 1,
        valueType: 'Category',
        majorGridLines: { width: 0 }, minorGridLines: { width: 0 },
        majorTickLines: { width: 0 }, minorTickLines: { width: 0 },
        lineStyle: { width: 0 },
        labelIntersectAction: 'Rotate90'
      };
      this.primaryYAxis = {
        title: 'Frequency of Occurrence',
        minimum: 0,
        maximum: 25,
        interval: 5,
        lineStyle: { width: 0 },
        majorTickLines: { width: 0 }, majorGridLines: { width: 1 },
        minorGridLines: { width: 1 }, minorTickLines: { width: 0 }
      };
      this.tooltip = { enable: true, shared: true, format: '${series.name}
: <b>${point.y}</b>' };
      this.title = 'Defects in Shirts';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See also

- [Data label](#)
- [Tooltip](#)

Chart series in Angular Chart component

Multiple Series

You can add multiple series to the chart by using [series](#) property.

The series are rendered in the order as it is added to the series array.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'

```

```

import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='gold'></e-series>
    <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver'></e-series>
    <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = [
      { country: "USA", gold: 50, silver: 70, bronze: 45 },
      { country: "China", gold: 40, silver: 60, bronze: 55 },
      { country: "Japan", gold: 70, silver: 60, bronze: 50 },
      { country: "Australia", gold: 60, silver: 56, bronze: 40 },
      { country: "France", gold: 50, silver: 45, bronze: 35 },
      { country: "Germany", gold: 40, silver: 30, bronze: 22 },
      { country: "Italy", gold: 40, silver: 35, bronze: 37 },
      { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
    ];
    this.primaryXAxis = {
      valueType: 'Category',
      title: 'Countries'
    };
    this.primaryYAxis = {
      minimum: 0, maximum: 80,
      interval: 20, title: 'Medals'
    };
    this.title = 'Olympic Medals';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';

```

```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Combination Series

Combination of different types like Line, column etc, can be rendered in a chart.

Bar series cannot be combined with any other series as the axis orientation is different from other series.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService } from '@syncfusion/ej2-
angular-charts'
import { StackingColumnSeriesService, LineSeriesService } from
'@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, ColumnSeriesService,
    StackingColumnSeriesService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='StackingColumn'
xName='x' yName='y' name='Private Consumption'></e-series>
    <e-series [dataSource]='chartData' type='StackingColumn'
xName='x' yName='y1' name='Government Consumption'></e-series>
    <e-series [dataSource]='chartData' type='StackingColumn'
xName='x' yName='y2' name='Investment'></e-series>
    <e-series [dataSource]='chartData' type='StackingColumn'
xName='x' yName='y3' name='Net Foreign Trade'></e-series>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y4' name='GDP' [marker]='marker' opacity=0.6></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.chartData = [
      { x: '2005', y: 1.2, y1: 0.5, y2: 0.7, y3: -0.8, y4: 1.5 },
      { x: '2006', y: 1, y1: 0.5, y2: 1.4, y3: 0, y4: 2.3 },
      { x: '2007', y: 1, y1: 0.5, y2: 1.5, y3: -1, y4: 2 },
      { x: '2008', y: 0.25, y1: 0.35, y2: 0.35, y3: -.35, y4: 0.1 },
```

```

        { x: '2009', y: 0.1, y1: 0.9, y2: -2.7, y3: -0.3, y4: -2.7 },
        { x: '2010', y: 1, y1: 0.5, y2: 0.5, y3: -0.5, y4: 1.8 },
        { x: '2011', y: 0.1, y1: 0.25, y2: 0.25, y3: 0, y4: 2 },
        { x: '2012', y: -0.25, y1: -0.5, y2: -0.1, y3: -0.4, y4: 0.4
    },

        { x: '2013', y: 0.25, y1: 0.5, y2: -0.3, y3: 0, y4: 0.9 },
        { x: '2014', y: 0.6, y1: 0.6, y2: -0.6, y3: -0.6, y4: 0.4 },
        { x: '2015', y: 0.9, y1: 0.5, y2: 0, y3: -0.3, y4: 1.3 }
    ];
    this.primaryXAxis = {
        title: 'Years',
        interval: 1,
        labelIntersectAction : 'Rotate45',
        valueType: 'Category'
    };
    this.primaryYAxis = {
        title: 'Growth',
        minimum: -3, maximum: 3, interval: 1
    };
    this.marker = { visible: true, width: 10, opacity: 0.6, height: 10
};
    this.title = 'Annual Growth GDP in France';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Enable Complex Property in Series

By setting `enableComplexProperty` value as `true` in series you can bind complex data to the chart.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService } from '@syncfusion/ej2-
angular-charts'
import { StackingColumnSeriesService, LineSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ CategoryService, ColumnSeriesService,
        StackingColumnSeriesService, LineSeriesService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart style='display:block;' align='center' id='chart-
container' [primaryXAxis]='primaryXAxis' [title]='title'>

```

```

        <e-series-collection>
            <e-series [dataSource]='data' type='Column' xName='group.x'
yName='group.y' name='Gold' width=2 enableComplexProperty=true> </e-series>
            <e-series [dataSource]='data' type='Column' xName='group.x'
yName='y' name='Silver' width=2 enableComplexProperty=true> </e-series>
        </e-series-collection>
    </ejs-chart>`
    })
    export class AppComponent implements OnInit {
        ngOnInit(): void {
        }
        public data: Object[] = [
            {group: { x: 'USA', y: 10}, y: 20},
            {group: { x: 'China', y: 30}, y: 10}
        ];
        //Initializing Primary X Axis
        public primaryXAxis: Object = {
            valueType: 'Category'
        };
        public title: string = 'Olympic Medal Counts - RIO';
    }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

Technical indicators in Angular Chart component

A [technical indicator](#) is a mathematical calculation based on historic price, volume or open interest information that aims to forecast financial market direction.

Chart supports 10 types of technical indicators.

Accumulation Distribution

Accumulation Distribution combines price and volume to show how money may be flowing into or out of stock.

To render a Accumulation Distribution Indicator, use indicator [type](#) as `AccumulationDistribution` and inject `AccumulationDistributionIndicatorService` into the `@NgModule.providers`.

To calculate the signal line [volume](#) field is additionally added with `dataSource`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService,
AccumulationDistributionIndicatorService, DateTimeService } from
'@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import { IAxisLabelRenderEventArgs } from '@syncfusion/ej2-angular-charts';

```

```

let chartData: any[] = [
  {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
  {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
  {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
  {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
  {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
  {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
  {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
  {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
  {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
  {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
  {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
  {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
  {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
  {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
  {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
  {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
  {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
  {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
  {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
  {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
  {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
  {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
  {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
  {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
  {x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
  {x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
  {x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
  {x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},

```



```

    {x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
    {x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
    {x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
    {x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},

```

```

    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
@Component({
imports: [
    ChartModule
],
providers: [ CandleSeriesService, LineSeriesService,
AccumulationDistributionIndicatorService, DateTimeService],
standalone: true,
selector: 'app-container',
template:
`<ejs-chart id='chartcontainer' style="display:block;"
[title]='title' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[axes] = 'axes' [crosshair] = 'crosshair' (axisLabelRender)=
'axisLabelRender($event)' [chartArea]= 'chartArea'>
    <e-series-collection>
        <e-series [dataSource]='data' type='Candle' xName='x'
high='high' low='low' open='open' close='close' volume='volume' name='Apple
Inc'> </e-series>
    </e-series-collection>
    <e-indicators>
        <e-indicator type='AccumulationDistribution' xName='x'
field="Close" yAxisName='secondary' fill="blue" seriesName='Apple Inc'> </e-
indicator>
    </e-indicators>
</ejs-chart>`
})
export class AppComponent implements OnInit {
    public data: Object[] = chartData;
    public primaryXAxis: Object = {
        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0}
    };
    public primaryYAxis: Object = {
        title: 'Price',
        labelFormat: '${value}',
        minimum: 30, maximum: 180,
        interval: 30,
    };
    public axes: Object = [{
        name: 'secondary',
        opposedPosition: true,
        majorGridLines: { width: 0 },
    }];
    public title: string = 'AAPL 2012-2017';
    public chartArea : Object = {
        border: { width : 0}
    };
    crosshair: any;
    public axisLabelRender(args: IAxisLabelRenderEventArgs): void {
        if (args.axis.name === 'secondary') {
            let value: number = parseInt(args.text) / 1000000000;
            args.text = String(value) + 'bn';
        }
    }
}

```

```
};
constructor() {
  //code
}
ngOnInit(): void {
  throw new Error('Method not implemented.');
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Average True Range (ATR)

ATR measures the stock volatility by comparing the current value with the previous value. To render a Average True Range (ATR) Indicator, use indicator [type](#) as `Atr` and inject `AtrIndicatorService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService, AtrIndicatorService,
DateTimeService, CrosshairService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let chartData: Object[] = [
  {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
  {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
  {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
  {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
  {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
  {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
  {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
  {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
  {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
  {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
  {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
  {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
```

```

    {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
    {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
    {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
    {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
    {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
    {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
    {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
    {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
    {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
    {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
    {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
    {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
    {x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
    {x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
    {x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
    {x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
    {x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
    {x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
    {x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
    {x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},

```

```

    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CandleSeriesService, LineSeriesService, AtrIndicatorService,
DateTimeService, CrosshairService],
  standalone: true,
  selector: 'app-container',
  template:
`<ejs-chart id='chartcontainer' style="display:block;"
[title]='title' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[axes] = 'axes' [chartArea]= 'chartArea'
[crosshair]='crosshair' >
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Candle' xName='x'
high='high' low='low' open='open' close='close' volume='volume' name='Apple
Inc'> </e-series>
    </e-series-collection>
    <e-indicators>
      <e-indicator type='Atr' xName='x' field="Close"
yAxisName='secondary' fill="blue" [period]='period' seriesName='Apple Inc'>
</e-indicator>

```

```

        </e-indicators>
      </ejs-chart>`
    })
    export class AppComponent implements OnInit {
      public primaryXAxis?: Object;
      public primaryYAxis?: Object;
      public axes?: Object[];
      chartData: Object[] | any;
      public data: Object[] = chartData;
      public crosshair?: Object;
      title: any = 'AAPL 2012-2017';
      chartArea: any;
      period: any;
      ngOnInit(): void {
        this.chartData = chartData;
        this.crosshair = { enable: true };
        this.primaryXAxis = {
          title: 'Months',
          valueType: 'DateTime',
          intervalType: 'Months',
          majorGridLines: { width: 0 },
          crosshairTooltip: { enable: true },
        };
        this.primaryYAxis = {
          title: 'Price',
          labelFormat: '${value}',
          minimum: 30, maximum: 180,
          interval: 30,
          crosshairTooltip: { enable: true },
        };
        this.axes = [{
          name: 'secondary',
          opposedPosition: true,
          majorGridLines: { width: 0 },
        }];
      }
    }
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Bollinger Band

A chart overlay that shows the upper and lower limits of normal price movements based on the standard deviation of prices.

To render a Bollinger Band, use indicator [type](#) as `BollingerBand` and inject `BollingerBandsService` into the `@NgModule.providers`.

Bollinger band will be represented by three lines (upperLine, lowerLine, signalLine). Bollinger Band default values of the [period](#) is 14 and [standardDeviations](#) is 2.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService, BollingerBandsService,
DateTimeService, RangeAreaSeriesService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let chartData: any[] = [
    {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
    {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
    {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
    {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
    {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
    {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
    {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
    {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
    {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
    {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
    {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
    {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
    {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
    {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
    {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
    {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
    {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
    {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
    {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
    {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
    {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
    {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
    {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},

```

```

    {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
    {x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
    {x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
    {x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
    {x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
    {x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
    {x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
    {x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
    {x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},

```



```

    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
@Component({
imports: [
    ChartModule
],
providers: [ CandleSeriesService, LineSeriesService, BollingerBandsService,
DateTimeService, RangeAreaSeriesService],
standalone: true,
    selector: 'app-container',
    template:
        `<ejs-chart id='chartcontainer' style="display:block;"
[title]='title' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[chartArea]= 'chartArea'>
        <e-series-collection>
            <e-series [dataSource]='data' type='Candle' xName='x'
high='high' low='low' open='open' close='close' volume='volume' name='Apple
Inc'> </e-series>
        </e-series-collection>
        <e-indicators>
            <e-indicator type='BollingerBands' xName='x' field="Close"
fill="blue" seriesName='Apple Inc'> </e-indicator>
        </e-indicators>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
    public data: Object[] = chartData;
    public primaryXAxis: Object = {
        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0},
    };
    public primaryYAxis: Object = {
        title: 'Price',
        labelFormat: '${value}',
        minimum: 30, maximum: 180,
        interval: 30,
    };
    public title: string = 'AAPL 2012-2017';
    public chartArea : Object = {
        border: { width : 0}
    };
    constructor() {
        //code
    }
    ngOnInit(): void {

```

```

        throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customization of BollingerBand

stroke, stroke-width, and color of upperLine can be customized by using [upperLine](#), and the lowerLine can be customized by using [lowerLine](#) properties of indicator.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService, BollingerBandsService,
DateTimeService, RangeAreaSeriesService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let chartData: any[] = [
    {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
    {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
    {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
    {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
    {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
    {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
    {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
    {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
    {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
    {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
    {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
    {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
    {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
    {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
```

```

    {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
    {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
    {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
    {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
    {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
    {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
    {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
    {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
    {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
    {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
    {x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
    {x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
    {x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
    {x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
    {x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
    {x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
    {x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
    {x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},

```

```

    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
@Component({
imports: [
    ChartModule
],
providers: [ CandleSeriesService, LineSeriesService, BollingerBandsService,
DateTimeService, RangeAreaSeriesService],
standalone: true,
selector: 'app-container',
template:
`<ejs-chart id='chartcontainer' style="display:block;"
[title]='title' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[chartArea]= 'chartArea'>
    <e-series-collection>
        <e-series [dataSource]='data' type='Candle' xName='x'
high='high' low='low' open='open' close='close' volume='volume' name='Apple
Inc'> </e-series>
    </e-series-collection>
    <e-indicators>
        <e-indicator type='BollingerBands' xName='x' field="Close"
fill="blue" [period]='period' seriesName='Apple Inc'> </e-indicator>
    </e-indicators>
</ejs-chart>`
})
export class AppComponent implements OnInit {
    public data: Object[] = chartData;
    public primaryXAxis: Object = {

```

```

        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0},
    };
    public primaryYAxis: Object = {
        title: 'Price',
        labelFormat: '${value}',
        minimum: 30, maximum: 180,
        interval: 30,
    };
    public title: string = 'AAPL 2012-2017';
    public chartArea : Object = {
        border: { width : 0}
    };
    public period: number = 3;
    public crosshair: Object = {
        enable: true, lineType: 'Vertical'
    };
    constructor() {
        //code
    }
    ngOnInit(): void {
        throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Exponential Moving Average (EMA)

Moving average Indicators are used to define the direction of the trend. To render a EMA Indicator, use indicator [type](#) as `Ema` and inject `EMAIndicatorService` into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService, EmaIndicatorService,
DateTimeService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let chartData: any[] = [
    {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
    {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
    {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
    {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
```

```

    {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
    {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
    {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
    {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
    {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
    {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
    {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
    {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
    {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
    {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
    {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
    {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
    {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
    {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
    {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
    {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
    {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
    {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
    {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
    {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
    {x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
    {x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
    {x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
    {x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
    {x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
    {x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
    {x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
    {x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},

```

```

    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
  ];
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CandleSeriesService, LineSeriesService, EmaIndicatorService,
DateTimeService],

```

```

standalone: true,
  selector: 'app-container',
  template:
    `<ejs-chart id='chartcontainer' style="display:block;"
    [title]='title' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
    [chartArea]= 'chartArea'>
      <e-series-collection>
        <e-series [dataSource]='data' type='Candle' xName='x'
        high='high' low='low' open='open' close='close' volume='volume' name='Apple
        Inc'> </e-series>
      </e-series-collection>
      <e-indicators>
        <e-indicator type='Ema' xName='x' field="Close" fill="blue"
        seriesName='Apple Inc'> </e-indicator>
      </e-indicators>
    </ejs-chart>`
  })
export class AppComponent implements OnInit {
  public data: Object[] = chartData;
  public primaryXAxis: Object = {
    title: 'Months',
    valueType: 'DateTime',
    intervalType: 'Months',
    majorGridLines: { width: 0 },
  };
  public primaryYAxis: Object = {
    title: 'Price',
    labelFormat: '${value}',
    minimum: 30, maximum: 180,
    interval: 30,
  };
  public title: string = 'AAPL 2012-2017';
  public chartArea : Object = {
    border: { width : 0 }
  };
  constructor() {
    //code
  }
  ngOnInit(): void {
    throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Momentum

Momentum shows the speed at which the price of the stock is changing. To render a Momentum indicator, use indicator [type](#) as `Momentum` and inject `MomentumIndicatorService` into the

@NgModule.providers. Momentum indicator will be represented by two lines (upperLine, signalLine).In momentum indicator the upperBand value is always render at the value 100.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService, MomentumIndicatorService,
DateTimeService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let chartData: any[] = [
  {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
  {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
  {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
  {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
  {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
  {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
  {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
  {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
  {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
  {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
  {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
  {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
  {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
  {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
  {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
  {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
  {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
  {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
  {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
  {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
  {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
  {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
```

```

    {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
    {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
    {x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
    {x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
    {x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
    {x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
    {x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
    {x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
    {x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
    {x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},

```

```

    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
@Component({
imports: [
    ChartModule
],
providers: [ CandleSeriesService, LineSeriesService,
MomentumIndicatorService, DateTimeService],
standalone: true,
    selector: 'app-container',
    template:
        `<ejs-chart id='chartcontainer' style="display:block;"
[title]='title' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[chartArea]= 'chartArea'>
        <e-series-collection>
            <e-series [dataSource]='data' type='Candle' xName='x'
high='high' low='low' open='open' close='close' volume='volume' name='Apple
Inc'> </e-series>
        </e-series-collection>
        <e-indicators>
            <e-indicator type='Momentum' xName='x' field="Close"
fill="blue" seriesName='Apple Inc'> </e-indicator>
        </e-indicators>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
    public data: Object[] = chartData;
    public primaryXAxis: Object = {
        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0},
    };
    public primaryYAxis: Object = {
        title: 'Price',
        labelFormat: '${value}',
        minimum: 30, maximum: 180,
        interval: 30,
    };
    public title: string = 'AAPL 2012-2017';
    public chartArea : Object = {
        border: { width : 0}
    };
    constructor() {

```

```
//code
}ngOnInit(): void {
    throw new Error('Method not implemented.');
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customization of MomentumIndicator

stroke, stroke-width, and color of upperLine can be customized by using, [upperLine](#), property of indicator.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService, MomentumIndicatorService,
DateTimeService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let chartData: any[] = [
    {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
    {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
    {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
    {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
    {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
    {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
    {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
    {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
    {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
    {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
    {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
    {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
    {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
    {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
```

```

    {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
    {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
    {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
    {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
    {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
    {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
    {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
    {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
    {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
    {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
    {x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
    {x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
    {x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
    {x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
    {x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
    {x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
    {x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
    {x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},

```

```

    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
@Component({
imports: [
    ChartModule
],
providers: [ CandleSeriesService, LineSeriesService,
MomentumIndicatorService, DateTimeService],
standalone: true,
selector: 'app-container',
template:
`<ejs-chart id='chartcontainer' style="display:block;"
[title]='title' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[chartArea]= 'chartArea'>
    <e-series-collection>
        <e-series [dataSource]='data' type='Candle' xName='x'
high='high' low='low' open='open' close='close' volume='volume' name='Apple
Inc'> </e-series>
    </e-series-collection>
    <e-indicators>
        <e-indicator type='Momentum' xName='x' field="Close"
fill="blue" seriesName='Apple Inc'> </e-indicator>
    </e-indicators>
</ejs-chart>`
})
export class AppComponent implements OnInit {
    public data: Object[] = chartData;
    public primaryXAxis: Object = {

```

```

        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0},
    };
    public primaryYAxis: Object = {
        title: 'Price',
        labelFormat: '${value}',
        minimum: 30, maximum: 180,
        interval: 30,
    };
    public title: string = 'AAPL 2012-2017';
    public chartArea : Object = {
        border: { width : 0}
    };
    constructor() {
        //code
    }
    ngOnInit(): void {
        throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Moving Average Convergence Divergence (MACD)

MACD is based on the difference between two EMA's. To render a MACD Indicator, use indicator [type](#) as `MACD` and inject `MACDIndicatorService` into the `@NgModule.providers`. MACD indicator will be represented by MACD line, signal line, MACD histogram. MACD histogram is used to differentiate MACD line and signal line.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService, MacdIndicatorService,
DateTimeService, ColumnSeriesService} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let chartData: any[] = [
    {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
    {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
    {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
    {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
    {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
```

```

    {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
    {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
    {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
    {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
    {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
    {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
    {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
    {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
    {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
    {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
    {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
    {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
    {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
    {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
    {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
    {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
    {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
    {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
    {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
    {x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
    {x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
    {x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
    {x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
    {x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
    {x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
    {x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
    {x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},

```



```

    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CandleSeriesService, LineSeriesService, MacdIndicatorService,
DateTimeService, ColumnSeriesService],
  standalone: true,
  selector: 'app-container',

```

```

    template:
      `<ejs-chart id='chartcontainer' style="display:block;"
        [title]='title' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
          [chartArea]= 'chartArea'>
        <e-series-collection>
          <e-series [dataSource]='data' type='Candle' xName='x'
            high='high' low='low' open='open' close='close' volume='volume' name='Apple
            Inc'> </e-series>
        </e-series-collection>
        <e-indicators>
          <e-indicator type='Macd' xName='x' field="Close" fill="blue"
            seriesName='Apple Inc'> </e-indicator>
        </e-indicators>
      </ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public data: Object[] = chartData;
    public primaryXAxis: Object = {
      title: 'Months',
      valueType: 'DateTime',
      intervalType: 'Months',
      majorGridLines: { width: 0 },
    };
    public primaryYAxis: Object = {
      title: 'Price',
      labelFormat: '${value}',
      minimum: 30, maximum: 180,
      interval: 30,
    };
    public title: string = 'AAPL 2012-2017';
    public chartArea : Object = {
      border: { width : 0 }
    };
    constructor() {
      //code
    };
    ngOnInit(): void {
      throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customization of MACD

stroke, stroke-width, and color of macdLine can be customized by using [macdLine](#), property of indicator. The positive and negative changes of histogram can be customized by [macdPositiveColor](#) and [macdNegativeColor](#) properties. The [\[macdType\]](#) is used to define the type of MACD indicator. To customize the MACD period using [slowPeriod](#) and [fastPeriod](#) properties.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService, MacdIndicatorService,
DateTimeService, ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';

let chartData: any[] = [
    {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
    {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
    {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
    {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
    {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
    {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
    {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
    {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
    {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
    {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
    {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
    {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
    {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
    {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
    {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
    {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
    {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
    {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
    {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
    {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
    {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
    {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
    {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
    {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},

```

```

    {x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
    {x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
    {x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
    {x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
    {x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
    {x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
    {x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
    {x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},

```

```

        {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
        {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
        {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
        {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
        {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
    ];
    @Component({
    imports: [
        ChartModule
    ],
    providers: [ CandleSeriesService, LineSeriesService, MacdIndicatorService,
    DateTimeService, ColumnSeriesService],
    standalone: true,
        selector: 'app-container',
        template:
            `

```

```
;
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Relative Strength Index (RSI)

RSI shows how strongly a stock is moving in its current direction. To render a RSI Indicator, use indicator [type](#) as `Rsi` and inject `RsiIndicatorService` into the `@NgModule.providers`. RSI indicator will be represented by three lines (upperBand, lowerBand, signalLine). The upperBand and lowerBand values are customized by [overBought](#) and [overSold](#) properties of indicator and the signalLine is calculated by RSI formula.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService, RsiIndicatorService,
DateTimeService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let chartData: any[] = [
  {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
  {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
  {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
  {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
  {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
  {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
  {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
  {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
  {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
  {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
  {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
  {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
  {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
  {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
```

```

    {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
    {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
    {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
    {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
    {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
    {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
    {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
    {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
    {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
    {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
    {x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
    {x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
    {x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
    {x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
    {x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
    {x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
    {x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
    {x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},

```

```

    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
@Component({
imports: [
    ChartModule
],
providers: [ CandleSeriesService, LineSeriesService, RsiIndicatorService,
DateTimeService],
standalone: true,
selector: 'app-container',
template:
`<ejs-chart id='chartcontainer' style="display:block;"
[title]='title' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[chartArea]= 'chartArea'>
    <e-series-collection>
        <e-series [dataSource]='data' type='Candle' xName='x'
high='high' low='low' open='open' close='close' volume='volume' name='Apple
Inc'> </e-series>
    </e-series-collection>
    <e-indicators>
        <e-indicator type='Rsi' xName='x' field="Close" fill="blue"
seriesName='Apple Inc'> </e-indicator>
    </e-indicators>
</ejs-chart>`
})
export class AppComponent implements OnInit {
    public data: Object[] = chartData;
    public primaryXAxis: Object = {

```



```

        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0},
    };
    public primaryYAxis: Object = {
        title: 'Price',
        labelFormat: '${value}',
        minimum: 30, maximum: 180,
        interval: 30,
    };
    public title: string = 'AAPL 2012-2017';
    public chartArea : Object = {
        border: { width : 0}
    };
    constructor() {
        //code
    }
    ngOnInit(): void {
        throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Simple Moving Average (SMA)

Moving average Indicators are used to define the direction of the trend. To render a SMA Indicator, use indicator [type](#) as `Sma` and inject `SmaIndicatorService` module using `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService, SmaIndicatorService,
DateTimeService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let chartData: any[] = [
    {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
    {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
    {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
    {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
    {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
    {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
```

```

    {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
    {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
    {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
    {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
    {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
    {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
    {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
    {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
    {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
    {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
    {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
    {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
    {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
    {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
    {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
    {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
    {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
    {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
    {x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
    {x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
    {x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
    {x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
    {x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
    {x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
    {x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
    {x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},

```

```

    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CandleSeriesService, LineSeriesService, SmaIndicatorService,
DateTimeService],
  standalone: true,
  selector: 'app-container',
  template:

```

```

`<ejs-chart id='chartcontainer' style="display:block;"
[title]='title' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[chartArea]= 'chartArea'>
  <e-series-collection>
    <e-series [dataSource]='data' type='Candle' xName='x'
high='high' low='low' open='open' close='close' volume='volume' name='Apple
Inc'> </e-series>
  </e-series-collection>
  <e-indicators>
    <e-indicator type='Sma' xName='x' field="Close" fill="blue"
seriesName='Apple Inc'> </e-indicator>
  </e-indicators>
</ejs-chart>`
))
export class AppComponent implements OnInit {
  public data: Object[] = chartData;
  public primaryXAxis: Object = {
    title: 'Months',
    valueType: 'DateTime',
    intervalType: 'Months',
    majorGridLines: { width: 0}
  };
  public primaryYAxis: Object = {
    title: 'Price',
    labelFormat: '${value}',
    minimum: 30, maximum: 180,
    interval: 30,
  };
  public title: string = 'AAPL 2012-2017';
  public chartArea : Object = {
    border: { width : 0}
  };
  constructor() {
    //code
  }
  ngOnInit(): void {
    throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Stochastic

It shows how a stock is, when compared to previous state. To render a Stochastic indicator, use indicator [type](#) as `Stochastic` and inject `StochasticIndicatorService` module using `@NgModule.providers` method.

stochastic indicator will be represented by four lines (upperLine, lowerLine, periodLine, signalLine).

In stochastic indicator the upperBand value and lowerBand value is customized by [overBought](#) and [overBought](#) properties of indicators and the periodLine and signalLine is render based on stochastic formula.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService, StochasticIndicatorService,
DateTimeService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let chartData: any[] = [
  {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
  {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
  {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
  {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
  {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
  {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
  {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
  {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
  {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
  {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
  {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
  {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
  {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
  {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
  {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
  {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
  {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
  {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
  {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
  {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
  {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
  {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
```

```

    {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
    {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
    {x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
    {x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
    {x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
    {x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
    {x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
    {x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
    {x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
    {x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},

```

```

    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
@Component({
imports: [
    ChartModule
],
providers: [ CandleSeriesService, LineSeriesService,
StochasticIndicatorService, DateTimeService],
standalone: true,
    selector: 'app-container',
    template:
        `<ejs-chart id='chartcontainer' style="display:block;"
[title]='title' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[chartArea]= 'chartArea'>
        <e-series-collection>
            <e-series [dataSource]='data' type='Candle' xName='x'
high='high' low='low' open='open' close='close' volume='volume' name='Apple
Inc'> </e-series>
        </e-series-collection>
        <e-indicators>
            <e-indicator type='Stochastic' xName='x' field="Close"
fill="blue" [period]='period' seriesName='Apple Inc'> </e-indicator>
        </e-indicators>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
    public data: Object[] = chartData;
    public primaryXAxis: Object = {
        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0}
    };
    public primaryYAxis: Object = {
        title: 'Price',
        labelFormat: '${value}',
        minimum: 30, maximum: 180,
        interval: 30,
    };
    public title: string = 'AAPL 2012-2017';
    public chartArea : Object = {
        border: { width : 0}
    };
    period: any;

```

```

    constructor() {
        //code
    }
    ngOnInit(): void {
        throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customization of StochasticIndicator

stroke, stroke-width, and color of upperLine can be customized by using [upperLine](#), the lowerLine can be customized by using [lowerLine](#) and the periodLine can be customized by using [periodLine](#) properties of indicator. To customize the period to find the average price using [kPeriod](#) and [dPeriod](#) properties.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService, StochasticIndicatorService,
DateTimeService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';

let chartData: any[] = [
    {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
    {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
    {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
    {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
    {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
    {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
    {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
    {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
    {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
    {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
    {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
    {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
    {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
```



```

    {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
    {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
    {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
    {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
    {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
    {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
    {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
    {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
    {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
    {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
    {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
    {x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
    {x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
    {x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
    {x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
    {x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
    {x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
    {x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
    {x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},

```

```

    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
@Component({
imports: [
    ChartModule
],
providers: [ CandleSeriesService, LineSeriesService,
StochasticIndicatorService, DateTimeService],
standalone: true,
    selector: 'app-container',
    template:
`<ejs-chart id='chartcontainer' style="display:block;"
[title]='title' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[chartArea]= 'chartArea'>
    <e-series-collection>
        <e-series [dataSource]='data' type='Candle' xName='x'
high='high' low='low' open='open' close='close' volume='volume' name='Apple
Inc'> </e-series>
    </e-series-collection>
    <e-indicators>
        <e-indicator type='Stochastic' xName='x' field="Close"
fill="blue" seriesName='Apple Inc'> </e-indicator>
    </e-indicators>
</ejs-chart>`
})
export class AppComponent implements OnInit {

```

```

public data: Object[] = chartData;
public primaryXAxis: Object = {
  title: 'Months',
  valueType: 'DateTime',
  intervalType: 'Months',
  majorGridLines: { width: 0}
};
public primaryYAxis: Object = {
  title: 'Price',
  labelFormat: '${value}',
  minimum: 30, maximum: 180,
  interval: 30,
};
public title: string = 'AAPL 2012-2017';
public chartArea : Object = {
  border: { width : 0}
};
constructor() {
  //code
}
ngOnInit(): void {
  throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Triangular Moving Average (TMA)

Moving average indicators are used to define the direction of the trend. To render a TMA Indicator, use indicator [type](#) as TMA and inject TmaIndicatorService module using @NgModule.providers.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService, TmaIndicatorService,
DateTimeService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let chartData: any[] = [
  {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
  {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
  {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
  {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
  {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
```

```

    {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
    {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
    {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
    {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
    {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
    {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
    {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
    {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
    {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
    {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
    {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
    {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
    {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
    {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
    {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
    {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
    {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
    {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
    {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
    {x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
    {x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
    {x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
    {x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
    {x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
    {x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
    {x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
    {x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},

```

```

    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CandleSeriesService, LineSeriesService, TmaIndicatorService,
DateTimeService],
  standalone: true,
  selector: 'app-container',

```

```

    template:
      `<ejs-chart id='chartcontainer' style="display:block;"
        [title]='title' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
          [chartArea]= 'chartArea'>
        <e-series-collection>
          <e-series [dataSource]='data' type='Candle' xName='x'
            high='high' low='low' open='open' close='close' volume='volume' name='Apple
            Inc'> </e-series>
        </e-series-collection>
        <e-indicators>
          <e-indicator type='Tma' xName='x' field="Close" fill="blue"
            seriesName='Apple Inc'> </e-indicator>
        </e-indicators>
      </ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public data: Object[] = chartData;
    public primaryXAxis: Object = {
      title: 'Months',
      valueType: 'DateTime',
      intervalType: 'Months',
      majorGridLines: { width: 0 }
    };
    public primaryYAxis: Object = {
      title: 'Price',
      labelFormat: '${value}',
      minimum: 30, maximum: 180,
      interval: 30,
    };
    public title: string = 'AAPL 2012-2017';
    public chartArea : Object = {
      border: { width : 0 }
    };
    constructor() {
      //code
    }
    ngOnInit(): void {
      throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customization of Technical Indicators

stroke, stroke-width, and color of signalLine can be customized by using fill, width and dashArray properties. and the period property is used to predict the data forecast calculations. The field value is used to compare the current price with previous price. It is applicable to Bollinger bands and moving averages. The showZones property is used to show/hides the overBought and overSold regions. It is applicable for RSI and stochastic indicators.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService, SmaIndicatorService,
DateTimeService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let chartData: any[] = [
    {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
    {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
    {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
    {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
    {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
    {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
    {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
    {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
    {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
    {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
    {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
    {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
    {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
    {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
    {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
    {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
    {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
    {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
    {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
    {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
    {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
    {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
    {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
    {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},

```

```

    {x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
    {x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
    {x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
    {x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
    {x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
    {x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
    {x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
    {x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},
    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},

```



```

        {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
        {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
        {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
        {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
        {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
    ];
    @Component({
    imports: [
        ChartModule
    ],
    providers: [ CandleSeriesService, LineSeriesService, SmaIndicatorService,
    DateTimeService],
    standalone: true,
    selector: 'app-container',
    template:
        `<ejs-chart id='chartcontainer' style="display:block;"
[title]='title' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[chartArea]= 'chartArea'>
        <e-series-collection>
            <e-series [dataSource]='data' type='Candle' xName='x'
high='high' low='low' open='open' close='close' volume='volume' name='Apple
Inc'> </e-series>
        </e-series-collection>
        <e-indicators>
            <e-indicator type='Sma' xName='x' field="Close" fill="blue"
seriesName='Apple Inc'> </e-indicator>
        </e-indicators>
    </ejs-chart>`
    })
    export class AppComponent implements OnInit {
    public data: Object[] = chartData;
    public primaryXAxis: Object = {
        title: 'Months',
        valueType: 'DateTime',
        intervalType: 'Months',
        majorGridLines: { width: 0}
    };
    public primaryYAxis: Object = {
        title: 'Price',
        labelFormat: '${value}',
        minimum: 30, maximum: 180,
        interval: 30,
    };
    public title: string = 'AAPL 2012-2017';
    public chartArea : Object = {
        border: { width : 0}
    };
    constructor() {
        //code
    }
    ngOnInit(): void {
        throw new Error('Method not implemented.');
```

```
;
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Data Source

Usually technical indicators are added along with a financial series. The [seriesName](#) represents the series, the data of which has to be analysed through indicators.

Technical indicators can also be added without series using [dataSource](#) property of indicator.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService, SmaIndicatorService,
DateTimeService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let chartData: any[] = [
    {x: new Date('2012-10-15'), open: 90.3357, high: 93.2557, low:
87.0885,close: 87.12,volume: 646996264},
    {x: new Date('2012-10-22'), open: 87.4885, high: 90.7685, low:
84.4285,close: 86.2857,volume: 866040680 },
    {x: new Date('2012-10-29'), open: 84.9828, high: 86.1428, low:
82.1071,close: 82.4,volume: 367371310},
    {x: new Date('2012-11-05'), open: 83.3593, high: 84.3914, low:
76.2457,close: 78.1514,volume: 919719846},
    {x: new Date('2012-11-12'), open: 79.1643, high: 79.2143, low:
72.25,close: 75.3825,volume: 894382149},
    {x: new Date('2012-11-19'), open: 77.2443, high: 81.7143, low:
77.1257,close: 81.6428,volume: 527416747},
    {x: new Date('2012-11-26'), open: 82.2714, high: 84.8928, low:
81.7514,close: 83.6114,volume: 646467974},
    {x: new Date('2012-12-03'), open: 84.8071, high: 84.9414, low:
74.09,close: 76.1785,volume: 980096264},
    {x: new Date('2012-12-10'), open: 75, high: 78.5085, low: 72.2257,close:
72.8277,volume: 835016110},
    {x: new Date('2012-12-17'), open: 72.7043, high: 76.4143, low:
71.6043,close: 74.19,volume: 726150329},
    {x: new Date('2012-12-24'), open: 74.3357, high: 74.8928, low:
72.0943,close: 72.7984,volume: 321104733},
    {x: new Date('2012-12-31'), open: 72.9328, high: 79.2857, low:
72.7143,close: 75.2857,volume: 540854882},
    {x: new Date('2013-01-07'), open: 74.5714, high: 75.9843, low:
73.6,close: 74.3285,volume: 574594262},
    {x: new Date('2013-01-14'), open: 71.8114, high: 72.9643, low:
69.0543,close: 71.4285,volume: 803105621},
    {x: new Date('2013-01-21'), open: 72.08, high: 73.57, low:
62.1428,close: 62.84,volume: 971912560},
```

```

    {x: new Date('2013-01-28'), open: 62.5464, high: 66.0857, low:
62.2657,close: 64.8028,volume: 656549587},
    {x: new Date('2013-02-04'), open: 64.8443, high: 68.4014, low:
63.1428,close: 67.8543,volume: 743778993},
    {x: new Date('2013-02-11'), open: 68.0714, high: 69.2771, low:
65.7028,close: 65.7371,volume: 585292366},
    {x: new Date('2013-02-18'), open: 65.8714, high: 66.1043, low:
63.26,close: 64.4014,volume: 421766997},
    {x: new Date('2013-02-25'), open: 64.8357, high: 65.0171, low:
61.4257,close: 61.4957,volume: 582741215},
    {x: new Date('2013-03-04'), open: 61.1143, high: 62.2043, low:
59.8571,close: 61.6743,volume: 632856539},
    {x: new Date('2013-03-11'), open: 61.3928, high: 63.4614, low:
60.7343,close: 63.38,volume: 572066981},
    {x: new Date('2013-03-18'), open: 63.0643, high: 66.0143, low:
63.0286,close: 65.9871,volume: 552156035},
    {x: new Date('2013-03-25'), open: 66.3843, high: 67.1357, low:
63.0886,close: 63.2371,volume: 390762517},
    {x: new Date('2013-04-01'), open: 63.1286, high: 63.3854, low:
59.9543,close: 60.4571,volume: 505273732},
    {x: new Date('2013-04-08'), open: 60.6928, high: 62.57, low:
60.3557,close: 61.4,volume: 387323550},
    {x: new Date('2013-04-15'), open: 61, high: 61.1271, low: 55.0143,close:
55.79,volume: 709945604},
    {x: new Date('2013-04-22'), open: 56.0914, high: 59.8241, low:
55.8964,close: 59.6007,volume: 787007506},
    {x: new Date('2013-04-29'), open: 60.0643, high: 64.7471, low: 60,close:
64.2828,volume: 655020017},
    {x: new Date('2013-05-06'), open: 65.1014, high: 66.5357, low:
64.3543,close: 64.71,volume: 545488533},
    {x: new Date('2013-05-13'), open: 64.5014, high: 65.4143, low:
59.8428,close: 61.8943,volume: 633706550},
    {x: new Date('2013-05-20'), open: 61.7014, high: 64.05, low:
61.4428,close: 63.5928,volume: 494379068},
    {x: new Date('2013-05-27'), open: 64.2714, high: 65.3, low:
62.7714,close: 64.2478,volume: 362907830},
    {x: new Date('2013-06-03'), open: 64.39, high: 64.9186, low:
61.8243,close: 63.1158,volume: 443249793},
    {x: new Date('2013-06-10'), open: 63.5328, high: 64.1541, low:
61.2143,close: 61.4357,volume: 389680092},
    {x: new Date('2013-06-17'), open: 61.6343, high: 62.2428, low:
58.3,close: 59.0714,volume: 400384818},
    {x: new Date('2013-06-24'), open: 58.2, high: 58.38, low: 55.5528,close:
56.6471,volume: 519314826},
    {x: new Date('2013-07-01'), open: 57.5271, high: 60.47, low:
57.3171,close: 59.6314,volume: 343878841},
    {x: new Date('2013-07-08'), open: 60.0157, high: 61.3986, low:
58.6257,close: 60.93,volume: 384106977},
    {x: new Date('2013-07-15'), open: 60.7157, high: 62.1243, low:
60.5957,close: 60.7071,volume: 286035513},
    {x: new Date('2013-07-22'), open: 61.3514, high: 63.5128, low:
59.8157,close: 62.9986,volume: 395816827},
    {x: new Date('2013-07-29'), open: 62.9714, high: 66.1214, low:
62.8857,close: 66.0771,volume: 339668858},
    {x: new Date('2013-08-12'), open: 65.2657, high: 72.0357, low:
65.2328,close: 71.7614,volume: 711563584},

```

```

    {x: new Date('2013-08-19'), open: 72.0485, high: 73.3914, low:
71.1714,close: 71.5743,volume: 417119660},
    {x: new Date('2013-08-26'), open: 71.5357, high: 72.8857, low:
69.4286,close: 69.6023,volume: 392805888},
    {x: new Date('2013-09-02'), open: 70.4428, high: 71.7485, low:
69.6214,close: 71.1743,volume: 317244380},
    {x: new Date('2013-09-09'), open: 72.1428, high: 72.56, low:
66.3857,close: 66.4143,volume: 669376320},
    {x: new Date('2013-09-16'), open: 65.8571, high: 68.3643, low:
63.8886,close: 66.7728,volume: 625142677},
    {x: new Date('2013-09-23'), open: 70.8714, high: 70.9871, low:
68.6743,close: 68.9643,volume: 475274537},
    {x: new Date('2013-09-30'), open: 68.1786, high: 70.3357, low:
67.773,close: 69.0043,volume: 368198906},
    {x: new Date('2013-10-07'), open: 69.5086, high: 70.5486, low:
68.3257,close: 70.4017,volume: 361437661},
    {x: new Date('2013-10-14'), open: 69.9757, high: 72.7514, low:
69.9071,close: 72.6985,volume: 342694379},
    {x: new Date('2013-10-21'), open: 73.11, high: 76.1757, low:
72.5757,close: 75.1368,volume: 490458997},
    {x: new Date('2013-10-28'), open: 75.5771, high: 77.0357, low:
73.5057,close: 74.29,volume: 508130174},
    {x: new Date('2013-11-04'), open: 74.4428, high: 75.555, low:
73.1971,close: 74.3657,volume: 318132218},
    {x: new Date('2013-11-11'), open: 74.2843, high: 75.6114, low:
73.4871,close: 74.9987,volume: 306711021},
    {x: new Date('2013-11-18'), open: 74.9985, high: 75.3128, low:
73.3814,close: 74.2571,volume: 282778778},
];
@Component({
imports: [
    ChartModule
],
providers: [ CandleSeriesService, LineSeriesService, SmaIndicatorService,
DateTimeService],
standalone: true,
    selector: 'app-container',
    template:
        `

```

```

        title: 'Price',
        labelFormat: '${value}',
        minimum: 30, maximum: 180,
        interval: 30,
    };
    public title: string = 'AAPL 2012-2017';
    public chartArea : Object = {
        border: { width : 0}
    };
    constructor() {
        //code
    }
    ngOnInit(): void {
        throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Module Dependencies

To render a Indicator, it is mandatory to inject `LineSeriesService` module..

In addition to that, MACD Indicator requires `ColumnSeriesService` and BollingerBands requires `RangeAreaSeriesService`.

<!-- markdownlint-disable MD036 -->

Trend lines in Angular Chart component

Trendlines are used to show the direction and speed of price.

Trendlines can be generated for Cartesian type series (Line, Column, Scatter, Area, Candle, Hilo etc.) except bar type series. You can add more than one trendline to a series.

Chart supports 6 types of trendlines.

Linear

A linear trendline is a best fit straight line that is used with simpler data sets. To render a linear trendline, use trendline [type](#) as `Linear` and inject `TrendLines`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let series1 : any[] =[];
let yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95,
```

```

        13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52, 36.36,
        41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
        43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
let point1; let i; let j = 0;
for (i = 1973; i <= 2013; i++) {
    point1 = { x: i, y: yValue[j] };
    series1.push(point1); j++;
}
@Component({
imports: [
    ChartModule
],
providers: [ ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService],
standalone: true,
selector: 'app-container',
template:
`<ejs-chart id='chartcontainer' [title]='title'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[chartArea]= 'chartArea'>
    <e-series-collection>
        <e-series [dataSource]='data' type='Scatter' xName='x'
yName='y' fill="#0066FF">
            <e-trendlines>
                <e-trendline type='Linear' width=3 name='Linear'
fill='#C64A75'>
                    </e-trendline>
            </e-trendlines>
        </e-series>
    </e-series-collection>
</ejs-chart>`
}))
export class AppComponent implements OnInit {
    public data: Object[] = series1;
    public primaryXAxis: Object = {
        title: 'Months',
        majorGridLines: { width : 0}
    };
    public primaryYAxis: Object = {
        title: 'Rupees against Dollars',
        interval: 10, lineStyle: {width: 0}, majorTickLines: { width: 0 }
    };
    public chartArea : Object = {
        border: { width : 0}
    };
    public title: string = 'Historical Indian Rupee Rate (INR USD)';
    constructor() {
        //code
    }
    ngOnInit(): void {
        throw new Error('Method not implemented.');
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Exponential

An exponential trendline is a curved line that is most useful when data values rise or fall at increasingly higher rates. You cannot create an exponential trendline, if your data contains zero or negative values.

To render a exponential trendline,

use trendline [type](#) as **Exponential** and inject

TrendLines.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let series1 : any[] =[];
let yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
let point1; let i; let j = 0;
for (i = 1973; i <= 2013; i++) {
    point1 = { x: i, y: yValue[j] };
    series1.push(point1); j++;
}
@Component({
    imports: [
        ChartModule
    ],
    providers: [ ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService ],
    standalone: true,
    selector: 'app-container',
    template:
        `<ejs-chart id='chartcontainer' [title]='title'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[chartArea]= 'chartArea'>
    <e-series-collection>
        <e-series [dataSource]='data' type='Scatter' xName='x'
yName='y' fill='#0066FF'>
            <e-trendlines>
                <e-trendline type='Linear' width=3
name='Exponential' fill='#C64A75'>
            </e-trendline>
            </e-trendlines>
        </e-series>
    </e-series-collection>
</ejs-chart>`
```

```

})
export class AppComponent implements OnInit {
  public data: Object[] = series1;
  public primaryXAxis: Object = {
    title: 'Months',
    majorGridLines: { width : 0}
  };
  public primaryYAxis: Object = {
    title: 'Rupees against Dollars',
    interval: 10, lineStyle: {width: 0}, majorTickLines: { width: 0 }
  };
  public chartArea : Object = {
    border: { width : 0}
  };
  public title: string = 'Historical Indian Rupee Rate (INR USD)';
  constructor() {
    //code
  }
  ngOnInit(): void {
    throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Logarithmic

A logarithmic trendline is a best-fit curved line that is most useful when the rate of change in the data increases or decreases quickly and then levels out.

A logarithmic trendline can use negative and/or positive values.

To render a logarithmic trendline, use trendline [type](#) as **Logarithmic** and inject **TrendLines**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let series1 : any[] =[];
let yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
let point1; let i; let j = 0;
for (i = 1973; i <= 2013; i++) {
  point1 = { x: i, y: yValue[j] };
  series1.push(point1); j++;
```



```

}
@Component({
  imports: [
    ChartModule
  ],
  providers: [ ScatterSeriesService, LineSeriesService, DateTimeService,
    TrendlinesService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-chart id='chartcontainer' [title]='title'
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
    [chartArea]= 'chartArea'>
      <e-series-collection>
        <e-series [dataSource]='data' type='Scatter' xName='x'
        yName='y' fill="#0066FF">
          <e-trendlines>
            <e-trendline type='Linear' width=3
            name='Logarithmic' fill='#C64A75'>
          </e-trendline>
          </e-trendlines>
        </e-series>
      </e-series-collection>
    </ejs-chart>`
  })
export class AppComponent implements OnInit {
  public data: Object[] = series1;
  public primaryXAxis: Object = {
    title: 'Months',
    majorGridLines: { width : 0}
  };
  public primaryYAxis: Object = {
    title: 'Rupees against Dollars',
    interval: 10, lineStyle: {width: 0}, majorTickLines: { width: 0 }
  };
  public chartArea : Object = {
    border: { width : 0}
  };
  public title: string = 'Historical Indian Rupee Rate (INR USD)';
  constructor() {
    //code
  }
  ngOnInit(): void {
    throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Polynomial

A polynomial trendline is a curved line that is used when data fluctuates.

To render a polynomial trendline, use trendline [type](#) as **Polynomial** and inject **TrendLines**.

polynomialOrder used to define the polynomial value.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let series1 : any[] =[];
let yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
let point1; let i; let j = 0;
for (i = 1973; i <= 2013; i++) {
    point1 = { x: i, y: yValue[j] };
    series1.push(point1); j++;
}
@Component({
imports: [
    ChartModule
],
providers: [ ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService],
standalone: true,
selector: 'app-container',
template:
`<ejs-chart id='chartcontainer' [title]='title'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[chartArea]= 'chartArea'>
    <e-series-collection>
        <e-series [dataSource]='data' type='Scatter' xName='x'
yName='y' fill='#0066FF'>
            <e-trendlines>
                <e-trendline type='Linear' width=3 name='Linear'
fill='#C64A75'>
            </e-trendline>
            </e-trendlines>
        </e-series>
    </e-series-collection>
</ejs-chart>`
}))
export class AppComponent implements OnInit {
    public data: Object[] = series1;
    public primaryXAxis: Object = {
        title: 'Months',
        majorGridLines: { width : 0}
    };
    public primaryYAxis: Object = {
```

```

        title: 'Rupees against Dollars',
        interval: 10, lineStyle: {width: 0}, majorTickLines: { width: 0 }
    };
    public chartArea : Object = {
        border: { width : 0}
    };
    public title: string = 'Historical Indian Rupee Rate (INR USD)';
    constructor() {
        //code
    }
    ngOnInit(): void {
        throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Power

A power trendline is a curved line that is best used with data sets that compare measurements that increase at a specific rate.

To render a power trendline, use trendline [type](#) as **Power** and inject

TrendLines.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let series1 : any[] =[];
let yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
let point1; let i; let j = 0;
for (i = 1973; i <= 2013; i++) {
    point1 = { x: i, y: yValue[j] };
    series1.push(point1); j++;
}
@Component({
    imports: [
        ChartModule
    ],
    providers: [ ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService],
    standalone: true,
```

```

    selector: 'app-container',
    template:
      `<ejs-chart id='chartcontainer' [title]='title'
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
      [chartArea]= 'chartArea'>
      <e-series-collection>
        <e-series [dataSource]='data' type='Scatter' xName='x'
yName='y' fill="#0066FF">
          <e-trendlines>
            <e-trendline type='Linear' width=3 name='Linear'
fill='#C64A75'>
              </e-trendline>
            </e-trendlines>
          </e-series>
        </e-series-collection>
      </ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public data: Object[] = series1;
    public primaryXAxis: Object = {
      title: 'Months',
      majorGridLines: { width : 0}
    };
    public primaryYAxis: Object = {
      title: 'Rupees against Dollars',
      interval: 10, lineStyle: {width: 0}, majorTickLines: { width: 0 }
    };
    public chartArea : Object = {
      border: { width : 0}
    };
    public title: string = 'Historical Indian Rupee Rate (INR USD)';
    constructor() {
      //code
    }
    ngOnInit(): void {
      throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Moving Average

A moving average trendline smoothen out fluctuations in data to show a pattern or trend more clearly.

To render a moving average trendline, use trendline [type](#) as `MovingAverage` and inject `TrendLines`.

`period` property defines the period to find the moving average.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
```

```

import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let series1 : any[] =[];
let yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
let point1; let i; let j = 0;
for (i = 1973; i <= 2013; i++) {
    point1 = { x: i, y: yValue[j] };
    series1.push(point1); j++;
}
@Component({
imports: [
    ChartModule
],
providers: [ ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService],
standalone: true,
selector: 'app-container',
template:
`<ejs-chart id='chartcontainer' [title]='title'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[chartArea]='chartArea'>
    <e-series-collection>
        <e-series [dataSource]='data' type='Scatter' xName='x'
yName='y' fill='#0066FF'>
            <e-trendlines>
                <e-trendline type='MovingAverage' width=3
name='Linear' fill='#C64A75'>
            </e-trendline>
            </e-trendlines>
        </e-series>
    </e-series-collection>
</ejs-chart>`
}))
export class AppComponent implements OnInit {
    public data: Object[] = series1;
    public primaryXAxis: Object = {
        title: 'Months',
        majorGridLines: { width : 0}
    };
    public primaryYAxis: Object = {
        title: 'Rupees against Dollars',
        interval: 10, lineStyle: {width: 0}, majorTickLines: { width: 0 }
    };
    public chartArea : Object = {
        border: { width : 0}
    };
    public title: string = 'Historical Indian Rupee Rate (INR USD)';
    constructor() {
        //code
    }
    ngOnInit(): void {

```

```

        throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customization of Trendlines

The [fill](#) and [width](#)

properties are used to customize the appearance of the trendline.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let series1 : any[] =[];
let yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
let point1; let i; let j = 0;
for (i = 1973; i <= 2013; i++) {
    point1 = { x: i, y: yValue[j] };
    series1.push(point1); j++;
}
@Component({
imports: [
    ChartModule
],
providers: [ ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService],
standalone: true,
selector: 'app-container',
template:
`<ejs-chart id='chartcontainer' [title]='title'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[chartArea]='chartArea'>
    <e-series-collection>
        <e-series [dataSource]='data' type='Scatter' xName='x'
yName='y' fill='#0066FF'>
            <e-trendlines>
                <e-trendline type='MovingAverage' width=3
name='Linear' fill='#C64A75'>
                    </e-trendline>
```

```

        </e-trendlines>
    </e-series>
</e-series-collection>
</ejs-chart>`
}))
export class AppComponent implements OnInit {
    public data: Object[] = series1;
    public primaryXAxis: Object = {
        title: 'Months',
        majorGridLines: { width : 0}
    };
    public primaryYAxis: Object = {
        title: 'Rupees against Dollars',
        interval: 10, lineStyle: {width: 0}, majorTickLines: { width: 0 }
    };
    public chartArea : Object = {
        border: { width : 0}
    };
    public title: string = 'Historical Indian Rupee Rate (INR USD)';
    constructor() {
        //code
    }
    ngOnInit(): void {
        throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Forecasting

Trendlines forecasting is the prediction of future/past situations.

Forecasting can be classified into two types as follows

Forward Forecasting

Backward Forecasting

Forward Forecasting

The value set for forwardForecast is used to determine the distance moving towards the future trend.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let series1 : any[] =[];
```

```

let yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
let point1; let i; let j = 0;
for (i = 1973; i <= 2013; i++) {
    point1 = { x: i, y: yValue[j] };
    series1.push(point1); j++;
}
@Component({
imports: [
    ChartModule
],
providers: [ ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService],
standalone: true,
selector: 'app-container',
template:
`<ejs-chart id='chartcontainer' [title]='title'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[chartArea]= 'chartArea'>
    <e-series-collection>
        <e-series [dataSource]='data' type='Scatter' xName='x'
yName='y' fill="#0066FF">
            <e-trendlines>
                <e-trendline type='MovingAverage' width=3
name='Linear' fill='#C64A75' [forwardForecast]='forwardForecast'>
            </e-trendline>
        </e-trendlines>
    </e-series>
</e-series-collection>
</ejs-chart>`
}))
export class AppComponent implements OnInit {
    public data: Object[] = series1;
    public primaryXAxis: Object = {
        title: 'Months',
        majorGridLines: { width : 0}
    };
    public primaryYAxis: Object = {
        title: 'Rupees against Dollars',
        interval: 10, lineStyle: {width: 0}, majorTickLines: { width: 0 }
    };
    public chartArea : Object = {
        border: { width : 0}
    };
    public forwardForecast: number = 5;
    public title: string = 'Historical Indian Rupee Rate (INR USD)';
    constructor() {
        //code
    }
    ngOnInit(): void {
        throw new Error('Method not implemented.');
```


MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Backward Forecasting

The value set for the backwardForecast is used to determine the past trends.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let series1 : any[] =[];
let yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
let point1; let i; let j = 0;
for (i = 1973; i <= 2013; i++) {
    point1 = { x: i, y: yValue[j] };
    series1.push(point1); j++;
}
@Component({
imports: [
    ChartModule
],
providers: [ ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService],
standalone: true,
selector: 'app-container',
template:
`<ejs-chart id='chartcontainer' [title]='title'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[chartArea]= 'chartArea'>
    <e-series-collection>
        <e-series [dataSource]='data' type='Scatter' xName='x'
yName='y' fill="#0066FF">
            <e-trendlines>
                <e-trendline type='MovingAverage' width=3
name='Linear' fill='#C64A75' [backwardForecast]='backwardForecast'>
            </e-trendline>
            </e-trendlines>
        </e-series>
    </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
```

```

public data: Object[] = series1;
public primaryXAxis: Object = {
    title: 'Months',
    majorGridLines: { width : 0}
};
public primaryYAxis: Object = {
    title: 'Rupees against Dollars',
    interval: 10, lineStyle: {width: 0}, majorTickLines: { width: 0 }
};
public chartArea : Object = {
    border: { width : 0}
};
public backwardForecast: number = 5;
public title: string = 'Historical Indian Rupee Rate (INR USD)';
constructor() {
    //code
}
ngOnInit(): void {
    throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Show or hide a trendline

You can show or hide the trendline by setting trendline **visible** property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
let series1 : any[] =[];
let yValue = [7.66, 8.03, 8.41, 8.97, 8.77, 8.20, 8.16, 7.89, 8.68, 9.48,
10.11, 11.36, 12.34, 12.60, 12.95,
13.91, 16.21, 17.50, 22.72, 28.14, 31.26, 31.39, 32.43, 35.52, 36.36,
41.33, 43.12, 45.00, 47.23, 48.62, 46.60, 45.28, 44.01, 45.17, 41.20,
43.41, 48.32, 45.65, 46.61, 53.34, 58.53];
let point1; let i; let j = 0;
for (i = 1973; i <= 2013; i++) {
    point1 = { x: i, y: yValue[j] };
    series1.push(point1); j++;
}
@Component({
    imports: [
        ChartModule
    ],
```

```

providers: [ ScatterSeriesService, LineSeriesService, DateTimeService,
TrendlinesService],
standalone: true,
  selector: 'app-container',
  template:
    `<ejs-chart id='chartcontainer' [title]='title'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[chartArea]= 'chartArea'>
    <e-series-collection>
      <e-series [dataSource]='data' type='Scatter' xName='x'
yName='y' fill="#0066FF">
        <e-trendlines>
          <e-trendline [visible]=false type='Linear' width=3
name='Linear' fill='#C64A75'>
        </e-trendline>
        </e-trendlines>
      </e-series>
    </e-series-collection>
  </ejs-chart>`
  ))
export class AppComponent implements OnInit {
  public data: Object[] = series1;
  public primaryXAxis: Object = {
    title: 'Months',
    majorGridLines: { width : 0}
  };
  public primaryYAxis: Object = {
    title: 'Rupees against Dollars',
    interval: 10, lineStyle: {width: 0}, majorTickLines: { width: 0 },
  };
  public chartArea : Object = {
    border: { width : 0}
  };
  public title: string = 'Historical Indian Rupee Rate (INR USD)';
  constructor() {
    //code
  }
  ngOnInit(): void {
    throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Data markers in Angular Chart component

Data markers are used to provide information about the data points in the series. You can add a shape to adorn each data point.

<!-- markdownlint-disable MD036 -->

Marker

<!-- markdownlint-disable MD036 -->

Markers can be added to points by enabling the [visible](#) option of the marker property. By default, distinct markers will be enabled for each series in the chart.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { markerData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='December 2007' width=2 [marker]='marker'></e-series>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y1' name='December 2008' width=2 [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = markerData;
    this.primaryXAxis = {
      valueType: 'Category', interval: 1,
    };
    this.marker = { visible: true };
    this.title = 'FB Penetration of Internet Audience';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Shape

Markers can be assigned with different shapes such as Rectangle, Circle, Diamond, etc. using the [shape](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { markerData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='December 2007' width=2 [marker]='marker'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = markerData;
    this.primaryXAxis = {
      valueType: 'Category'
    };
    this.marker = { visible: true, width: 10, height: 10, shape:
'Diamond' };
    this.title = 'FB Penetration of Internet Audience';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Note : To know more about the marker shape type refer the [shape](#).

Images

Apart from the shapes, you can also add custom images to mark the data point using the [imageUrl](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { imageData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='India' width=2 [marker]='marker'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = imageData;
    this.primaryXAxis = {
      valueType: 'Category'
    };
    this.marker = { visible: true,
      width: 10, height: 10, shape: 'Image',
      imageUrl: './sun_annotation.png'
    };
    this.title = 'Temperature flow over months';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customization

Marker's color and border can be customized using `fill` and `border` properties.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { markerData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='December 2007' width=2 [marker]='marker'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = markerData;
    this.primaryXAxis = {
      valueType: 'Category'
    };
    this.marker = { visible: true, fill: 'Red', height: 10, width: 10,
      border:{width: 2, color: 'blue'} };
    this.title = 'FB Penetration of Internet Audience';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customizing specific point

You can also customize the specific marker and label using `pointRender` event. The `pointRender` event allows you to change the shape, color and border for a point.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { markerData } from '../datasource';
import { IPointRenderEventArgs } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis'
(pointRender)='pointRender($event)' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='December 2007' width=2 [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  primaryYAxis: any;
  public pointRender(args: IPointRenderEventArgs): void {
    if(args.point.index === 3) {
      args.fill = 'red'
    }
  };
  ngOnInit(): void {
    this.chartData = markerData;
    this.primaryXAxis = {
      valueType: 'Category'
    };
    this.marker = { visible: true,
      height: 10, width: 10 };
    this.title = 'FB Penetration of Internet Audience';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```


Fill marker with series color

Marker can be filled with the series color by setting the [isFilled](#) property to `true`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { markerData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
(pointRender)='pointRender($event)' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='December 2007' width=2 [marker]='marker'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  pointRender($event: any) {
  }
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = markerData;
    this.primaryXAxis = {
      valueType: 'Category'
    };
    this.marker = { visible: true,
      height: 10, width: 10, isFilled: true };
    this.title = 'FB Penetration of Internet Audience';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See also

- [Customize the marker with different shape](#)

Data labels in Angular Chart component

Data label can be added to a chart series by enabling the [visible](#) option in the dataLabel. By default, the labels will arrange smartly without overlapping.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, CategoryService, LineSeriesService,
ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, DataLabelService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { columnData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, LegendService,
DataLabelService, ColumnSeriesService, CategoryService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Warmest' width=2 [marker]='marker'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.chartData = columnData;
    this.primaryXAxis = {
      valueType: 'Category'
    };
    this.marker = { dataLabel: { visible: true } };
    this.title = 'Alaska Weather Statistics - 2016';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
```

```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Note: To use datalabel feature, we need to inject `DataLabelService` into the `@NgModule.providers`.

Position

Using [position](#) property, you can place the label either on

Top, Middle, Bottom or **Outer** (outer is applicable for column and bar type series).

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, CategoryService, LineSeriesService,
ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, DataLabelService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { columnData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, LegendService,
DataLabelService, ColumnSeriesService, CategoryService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Warmest' width=2 [marker]='marker'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.chartData = columnData;
    this.primaryXAxis = {
      valueType: 'Category'
    };
    this.marker = { dataLabel: { visible: true, position: 'Middle' }
    };
    this.title = 'Alaska Weather Statistics - 2016';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Note: The position **Outer** is applicable for column and bar type series.

Data Label Template

Label content can be formatted by using the template option. Inside the template, you can add the placeholder text `${point.x}` and `${point.y}` to display corresponding data points x & y value.

Using [template](#) property, you can set data label template in chart.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, CategoryService, LineSeriesService,
ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, DataLabelService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { columnData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, LegendService,
DataLabelService, ColumnSeriesService, CategoryService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Warmest' width=2 [marker]='marker'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = columnData;
    this.primaryXAxis = {
      title: 'Months',
      valueType: 'Category', labelFormat: 'yMMM',
      edgeLabelPlacement: 'Shift'
    };
    this.marker = { dataLabel: { visible: true, position: 'Middle',
```

```

        template:
'<div>${point.x}</div><div>${point.y}</div>' }
    };
    this.title = 'Alaska Weather Statistics - 2016';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Format

Data label for the chart can be formatted using [format](#) property. You can use the global formatting options, such as 'n', 'p', and 'c'.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, CategoryService, LineSeriesService,
ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, DataLabelService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { columnData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, LegendService,
DataLabelService, ColumnSeriesService, CategoryService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Warmest' width=2 [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = columnData;
    this.primaryXAxis = {
      title: 'Months',

```

```

        valueType: 'Category', labelFormat: 'yMMM',
        edgeLabelPlacement: 'Shift'
    };
    this.marker = { dataLabel: { visible: true, format: 'p1' }
    };
    this.title = 'Alaska Weather Statistics - 2016';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Value	Format	Resultant Value	Description
1000	n1	1000.0	The number is rounded to 1 decimal place.
1000	n2	1000.00	The number is rounded to 2 decimal places.
1000	n3	1000.000	The number is rounded to 3 decimal place.
0.01	p1	1.0%	The number is converted to percentage with 1 decimal place.
0.01	p2	1.00%	The number is converted to percentage with 2 decimal place.
0.01	p3	1.000%	The number is converted to percentage with 3 decimal place.
1000	c1	\$1000.0	The currency symbol is appended to number and number is rounded to 1 decimal place.
1000	c2	\$1000.00	The currency symbol is appended to number and number is rounded to 2 decimal place.

Text Mapping

Text from the data source can be mapped using **name** property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, CategoryService, LineSeriesService,
ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, DataLabelService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { mapData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],

```

```

providers: [ DateTimeService, LineSeriesService, LegendService,
DataLabelService, ColumnSeriesService, CategoryService ],
standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Warmest' width=2 [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
}))
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.chartData = mapData;
    this.primaryXAxis = {
      valueType: 'Category'
    };
    this.marker = { dataLabel: { visible: true, name: 'text' }
    };
    this.title = 'Alaska Weather Statistics - 2016';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Margin

margin for data label can be applied to using left, right, bottom and top properties.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, CategoryService, LineSeriesService,
ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, DataLabelService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { columnData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],

```

```

providers: [ DateTimeService, LineSeriesService, LegendService,
DataLabelService, ColumnSeriesService, CategoryService ],
standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Warmest' width=2 [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
}))
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.chartData = columnData;
    this.primaryXAxis = {
      valueType: 'Category'
    };
    this.marker = { dataLabel: { visible: true, border: { width: 1, color :
'red'},
                    margin: {
                      left: 5,
                      right: 5,
                      top: 5,
                      bottom: 5
                    }
                  }
    };
    this.title = 'Alaska Weather Statistics - 2016';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

DataLabel Rotation

Using **angle** property, you can rotate the data label by its given angle.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, CategoryService, LineSeriesService,
ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, DataLabelService } from '@syncfusion/ej2-angular-
charts'

```



```

import { Component, OnInit } from '@angular/core';
import { columnData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, LegendService,
    DataLabelService, ColumnSeriesService, CategoryService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Warmest' width=2 [marker]='marker'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.chartData = columnData;
    this.primaryXAxis = {
      valueType: 'Category'
    };
    this.marker = { dataLabel: { visible: true, border:{width: 1, color
: 'red'},
      margin:{
        left:5,
        right:5,
        top:5,
        bottom:5
      }, angle : 45, enableRotation: true }
    };
    this.title = 'Alaska Weather Statistics - 2016';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customization

stroke and border of data label can be customized using fill and border properties. Rounded corners can be customized using rx and ry properties.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, CategoryService, LineSeriesService,
ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, DataLabelService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { columnData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, LegendService,
DataLabelService, ColumnSeriesService, CategoryService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Warmest' width=2 [marker]='marker'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.chartData = columnData;
    this.primaryXAxis = {
      valueType: 'Category'
    };
    this.marker = { dataLabel: { visible: true,
      border:{width: 2, color : 'red'},
      rx:10, ry: 10
    }
  };
    this.title = 'Alaska Weather Statistics - 2016';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note: **rx** and **ry** properties requires **border** values not to be null.

Customizing Specific Point

You can also customize the specific marker and label using [pointRender](#) and [textRender](#) event. **pointRender** event allows you to change the shape, color and border for a point, whereas the **textRender** event allows you to change the text for the point.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, CategoryService, LineSeriesService,
ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, DataLabelService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { IPointRenderEventArgs, ITextRenderEventArgs } from
 '@syncfusion/ej2-angular-charts';
import { columnData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, LegendService,
DataLabelService, ColumnSeriesService, CategoryService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis'
(pointRender)='pointRender($event)'
(textRender)='textRender($event)'[title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Warmest' width=2 [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  primaryYAxis: any;
  public pointRender(args: IPointRenderEventArgs): void {
    if(args.point.index === 6) {
      args.fill = 'red'
    }
  };
  public textRender(args: ITextRenderEventArgs): void {
    if(args.point.index === 6){
      args.text = 'Maximum Temperature';
      args.color = 'red';
    }
  };
  ngOnInit(): void {
    this.chartData = columnData;
    this.primaryXAxis = {
```

```

        title: 'Months',
        valueType: 'Category', labelFormat: 'yMMM',
        edgeLabelPlacement: 'Shift'
    };
    this.marker = { dataLabel: { visible: true }
    };
    this.title = 'Alaska Weather Statistics - 2016';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Show percentage based on each series points

You can calculate the percentage value based on the sum for each series using the `seriesRender` and `textRender` events in the chart. In `seriesRender` calculate the sum of each series y values and In `textRender` calculate percentage value based on the sum value and modify the text.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, CategoryService, LineSeriesService,
ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, DataLabelService } from '@syncfusion/ej2-angular-
charts'
import { Component, ViewEncapsulation } from '@angular/core';
import {
    ISeriesRenderEventArgs,
    ITextRenderEventArgs,
} from '@syncfusion/ej2-angular-charts';
import { Browser } from '@syncfusion/ej2-base';
let total: any = [];
/**
 * Sample for Column Series
 */
@Component({
    imports: [
        ChartModule
    ],
    providers: [ DateTimeService, LineSeriesService, LegendService,
DataLabelService, ColumnSeriesService, CategoryService ],
    standalone: true,
    selector: 'app-container',
    template: `<div class="control-section">
<div align="center">
    <ejs-chart style="display:block;" [chartArea]='chartArea'
[width]='width' align="center" id='chartcontainer'
[primaryXAxis]='primaryXAxis'

```

```

        [primaryYAxis]='primaryYAxis' [title]='title'
[tooltip]='tooltip' (seriesRender)='seriesRender($event)'
(textRender)='textRender($event) '>
        <e-series-collection>
            <e-series [dataSource]='data' type='Column' xName='x'
yName='y' name='Gold' width=2 [marker]='marker'> </e-series>
            <e-series [dataSource]='data1' type='Column' xName='x'
yName='y' name='Silver' width=2 [marker]='marker'> </e-series>
            <e-series [dataSource]='data2' type='Column' xName='x'
yName='y' name='Bronze' width=2 [marker]='marker'> </e-series>
        </e-series-collection>
    </ejs-chart>
</div>
<div style="float: right; margin-right: 10px; margin-top: -5px">Source:
    <a href="https://www.gov.uk/" target='_blank'>www.gov.uk</a>
</div>
</div>`,
    encapsulation: ViewEncapsulation.None,
})
export class AppComponent {
    public data: Object[] = [
        { x: 'USA', y: 46 },
        { x: 'GBR', y: 27 },
        { x: 'CHN', y: 26 },
    ];
    public data1: Object[] = [
        { x: 'USA', y: 37 },
        { x: 'GBR', y: 23 },
        { x: 'CHN', y: 18 },
    ];
    public data2: Object[] = [
        { x: 'USA', y: 38 },
        { x: 'GBR', y: 17 },
        { x: 'CHN', y: 26 },
    ];
    //Initializing Primary X Axis
    public primaryXAxis: Object = {
        valueType: 'Category',
        interval: 1,
        majorGridLines: { width: 0 },
    };
    //Initializing Primary Y Axis
    public primaryYAxis: Object = {
        majorGridLines: { width: 0 },
        majorTickLines: { width: 0 },
        lineStyle: { width: 0 },
        labelStyle: { color: 'transparent' },
    };
    public marker: Object = {
        dataLabel: {
            visible: true,
            position: 'Top',
            font: { fontWeight: '600', color: '#ffffff' },
        },
    };
    public title: string = 'Olympic Medal Counts - RIO';
    public tooltip: Object = {

```

```

        enable: true,
    };
    // custom code start
    public seriesRender(args: ISeriesRenderEventArgs | any): void {
        for (let i = 0; i < args.data.length; i++) {
            if (!total[args.data[i].x]) total[args.data[i].x] = 0;
            total[args.data[i].x] += parseInt(args.data[i].y);
        }
    }
    public textRender(args: ITextRenderEventArgs | any): void {
        let percentage: number | string = (parseInt(args.text) /
total[args.point.x]) * 100;
        percentage = percentage % 1 === 0 ? percentage :
percentage.toFixed(2);
        args.text = percentage + '%';
    }
    // custom code end
    public chartArea: Object = {
        border: {
            width: 0,
        },
    };
    public width: string = Browser.isDevice ? '100%' : '60%';
    constructor() {
        //code
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Show total stacking values in data label](#)
- [Prevent the data label when the data value is 0](#)

Chart annotations in Angular Chart component

Annotations are used to mark the specific area of interest in the chart area with texts, shapes or images.

<!-- markdownlint-disable MD033 -->

You can add annotations to the chart by using the `<annotations>` option. By using the

[Link to the Video](#) option of annotation object, you can specify

the id of the element that needs to be displayed in the chart area.

To know more about annotations, you can check on this video:

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'

```

```

import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
ColumnSeriesService, LineSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
imports: [
    ChartModule
],
providers: [ CategoryService, DateTimeService, ScrollBarService,
LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService,],
standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-annotations>
        <e-annotation content='70 Gold Medals' region='Series'
coordinateUnits='Point' x='Japan' y=75>
            </e-annotation>
        </e-annotations>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold'></e-series>
    </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    ngOnInit(): void {
        this.chartData = [
            { country: "USA", gold: 50 },
            { country: "China", gold: 40 },
            { country: "Japan", gold: 70 },
            { country: "Australia", gold: 60 },
            { country: "France", gold: 50 },
            { country: "Germany", gold: 40 },
            { country: "Italy", gold: 40 },
            { country: "Sweden", gold: 30 }
        ];
        this.primaryXAxis = {
            valueType: 'Category',
            title: 'Countries'
        };
        this.primaryYAxis = {
            minimum: 0, maximum: 80,
            interval: 20, title: 'Medals'
        };
        this.title = 'Olympic Medals';
    }
}

```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Note: To use annotation feature in chart, we need to inject **ChartAnnotationService** into the **@NgModule.providers**.

Region

Annotations can be placed either with respect to **Series** or **Chart**. by default, it will placed with respect to **Chart**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
  ColumnSeriesService, LineSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
  StackingColumnSeriesService, LegendService, TooltipService
  } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { columnData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
  LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
  StackingColumnSeriesService, LegendService, TooltipService, ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-annotations>
    <e-annotation content='<div>Highest Medal Count</div>'
region='Series' coordinateUnits='Point' x='Japan' y=75>
    </e-annotation>
  </e-annotations>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
```



```

public primaryYAxis?: Object;
ngOnInit(): void {
  this.chartData = columnData;
  this.primaryXAxis = {
    valueType: 'Category',
    title: 'Countries'
  };
  this.primaryYAxis = {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals'
  };
  this.title = 'Olympic Medals';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Co-ordinate Units

Specified the coordinates units of the annotation either **Pixel** or **Point**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
  ColumnSeriesService, LineSeriesService,
  ChartAnnotationService, RangeColumnSeriesService,
  StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { columnData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService, ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-annotations>
    <e-annotation content='<div style="border: 1px solid black;
padding: 5px 5px 5px 5px, background:#f5f5f5">Annotation in Pixel</div>'
    coordinateUnits='Pixel' x=150 y=75>
    </e-annotation>
  </e-annotations>
`
})

```

```

        <e-series-collection>
            <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold'></e-series>
        </e-series-collection>
    </ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public primaryYAxis?: Object;
        ngOnInit(): void {
            this.chartData = columnData;
            this.primaryXAxis = {
                valueType: 'Category',
                title: 'Countries'
            };
            this.primaryYAxis = {
                minimum: 0, maximum: 80,
                interval: 20, title: 'Medals'
            };
            this.title = 'Olympic Medals';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Alignment

Annotation provides `verticalAlignment` and `horizontalAlignment`. The `verticalAlignment` can be customized via `Top`, `Bottom` or `Middle` and the `horizontalAlignment` can be customized via `Near`, `Far` or `Center`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
ColumnSeriesService, LineSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService
    } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { columnData } from './datasource';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ CategoryService, DateTimeService, ScrollBarService,
LineSeriesService, ColumnSeriesService,

```

```

    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService,],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-annotations>
        <e-annotation content='<div style="border: 1px solid black;
padding: 5px 5px 5px 5px, background:#f5f5f5">Highest Medal Count</div>'
        verticalAlignment='Top' horizontalAlignment='Near' x='France'
y=50>
        </e-annotation>
    </e-annotations>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold'></e-series>
    </e-series-collection>
</ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    public border?: Object;
    public margin?: Object;
    ngOnInit(): void {
      this.chartData = columnData;
      this.primaryXAxis = {
        valueType: 'Category',
        title: 'Countries'
      };
      this.primaryYAxis = {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
      };
      this.title = 'Olympic Medals';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Adding y-axis sub title through on annotation

By setting text div in the **content** option of annotation object you can add sub title to chart y-axis. Specified the **coordinate** value as **pixel** and customize x and y location of the text.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'

```

```

import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
ColumnSeriesService, LineSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { columnData } from './datasource';
@Component({
imports: [
    ChartModule
],
providers: [ CategoryService, DateTimeService, ScrollBarService,
LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService,],
standalone: true,
selector: 'app-container',
template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-annotations>
        <e-annotation content='<div id="text" style="transform: rotate(-
90deg);">Speed</div>'
            verticalAlignment='Top' horizontalAlignment='Near' x=18 y=180
coordinateUnits='Pixel'>
            </e-annotation>
        </e-annotations>
        <e-series-collection>
            <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold'></e-series>
        </e-series-collection>
    </ejs-chart>
<style></style>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    public border?: Object;
    public margin?: Object;
    ngOnInit(): void {
        this.chartData = columnData;
        this.primaryXAxis = {
            valueType: 'Category',
            title: 'Countries'
        };
        this.primaryYAxis = {
            minimum: 0, maximum: 80,
            interval: 20, title: 'm3/min'
        };
        this.title = 'Olympic Medals';
    }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Show total stacking values in data label](#)
- [Create footer and watermark for chart](#)[Link to the Video](#)

Legend in Angular Chart component

Legend provides information about the series rendered in the chart.

To know more about legend settings, you can check on this video:

Position and Alignment

By using the [position](#) property, you can position the legend at left, right, top or bottom of the chart. The legend is positioned at the bottom of the chart, by default.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' ></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
```

```

public legendSettings?: Object;
ngOnInit(): void {
    this.chartData = [
        { country: "USA", gold: 50, silver: 70, bronze: 45 },
        { country: "China", gold: 40, silver: 60, bronze: 55 },
        { country: "Japan", gold: 70, silver: 60, bronze: 50 },
        { country: "Australia", gold: 60, silver: 56, bronze: 40 },
        { country: "France", gold: 50, silver: 45, bronze: 35 },
        { country: "Germany", gold: 40, silver: 30, bronze: 22 },
        { country: "Italy", gold: 40, silver: 35, bronze: 37 },
        { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
    ];
    this.primaryXAxis = {
        valueType: 'Category',
        title: 'Countries'
    };
    this.primaryYAxis = {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals',
    };
    this.legendSettings = { visible: true, position: 'Top' };
    this.title = 'Olympic Medals';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Custom position helps you to position the legend anywhere in the chart using x, y coordinates.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
    <e-series-collection>

```

```

        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver'></e-series>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' ></e-series>
    </e-series-collection>
</ejs-chart>`
    ))
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    public legendSettings?: Object;
    ngOnInit(): void {
        this.chartData = [
            { country: "USA", gold: 50, silver: 70, bronze: 45 },
            { country: "China", gold: 40, silver: 60, bronze: 55 },
            { country: "Japan", gold: 70, silver: 60, bronze: 50 },
            { country: "Australia", gold: 60, silver: 56, bronze: 40 },
            { country: "France", gold: 50, silver: 45, bronze: 35 },
            { country: "Germany", gold: 40, silver: 30, bronze: 22 },
            { country: "Italy", gold: 40, silver: 35, bronze: 37 },
            { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
        ];
        this.primaryXAxis = {
            valueType: 'Category',
            title: 'Countries'
        };
        this.primaryYAxis = {
            minimum: 0, maximum: 80,
            interval: 20, title: 'Medals',
        };
        this.legendSettings = {
            visible: true,
            position: 'Top',
            location: { x: 200, y: 40 }
        };
        this.title = 'Olympic Medals';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

Legend Reverse

<!-- markdownlint-disable MD036 -->

You can reverse the order of the legend items by using the [reverse](#) property. By default, legend for the first series in the collection will be placed first.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' ></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public legendSettings?: Object;
  ngOnInit(): void {
    this.chartData = [
      { country: "USA", gold: 50, silver: 70, bronze: 45 },
      { country: "China", gold: 40, silver: 60, bronze: 55 },
      { country: "Japan", gold: 70, silver: 60, bronze: 50 },
      { country: "Australia", gold: 60, silver: 56, bronze: 40 },
      { country: "France", gold: 50, silver: 45, bronze: 35 },
      { country: "Germany", gold: 40, silver: 30, bronze: 22 },
      { country: "Italy", gold: 40, silver: 35, bronze: 37 },
      { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
    ];
    this.primaryXAxis = {
      valueType: 'Category',
      title: 'Countries'
    };
    this.primaryYAxis = {
      minimum: 0, maximum: 80,

```



```

        interval: 20, title: 'Medals',
    };
    this.legendSettings = { visible: true, reverse: true };
    this.title = 'Olympic Medals';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Legend Alignment

<!-- markdownlint-disable MD036 -->

You can align the legend as **center**, **far** or **near** to the chart using [alignment](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' ></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;

```

```

public legendSettings?: Object;
ngOnInit(): void {
    this.chartData = [
        { country: "USA", gold: 50, silver: 70, bronze: 45 },
        { country: "China", gold: 40, silver: 60, bronze: 55 },
        { country: "Japan", gold: 70, silver: 60, bronze: 50 },
        { country: "Australia", gold: 60, silver: 56, bronze: 40 },
        { country: "France", gold: 50, silver: 45, bronze: 35 },
        { country: "Germany", gold: 40, silver: 30, bronze: 22 },
        { country: "Italy", gold: 40, silver: 35, bronze: 37 },
        { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
    ];
    this.primaryXAxis = {
        valueType: 'Category',
        title: 'Countries'
    };
    this.primaryYAxis = {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals',
    };
    this.legendSettings = { visible: true, position: 'Top', alignment:
'Near' };
    this.title = 'Olympic Medals';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customization

To change the legend icon shape, you can use [legendShape](#) property

in the [series](#). By default legend icon shape is `seriesType`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',

```

```

    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' legendShape='Circle'></e-series>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver' legendShape='SeriesType'></e-series>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' legendShape='Rectangle'></e-series>
    </e-series-collection>
    </ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    public legendSettings?: Object;
    ngOnInit(): void {
      this.chartData = [
        { country: "USA", gold: 50, silver: 70, bronze: 45 },
        { country: "China", gold: 40, silver: 60, bronze: 55 },
        { country: "Japan", gold: 70, silver: 60, bronze: 50 },
        { country: "Australia", gold: 60, silver: 56, bronze: 40 },
        { country: "France", gold: 50, silver: 45, bronze: 35 },
        { country: "Germany", gold: 40, silver: 30, bronze: 22 },
        { country: "Italy", gold: 40, silver: 35, bronze: 37 },
        { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
      ];
      this.primaryXAxis = {
        valueType: 'Category',
        title: 'Countries'
      };
      this.primaryYAxis = {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals',
      };
      this.legendSettings = { visible: true };
      this.title = 'Olympic Medals';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Legend Size

By default, legend takes 20% - 25% of the chart's height horizontally, when it is placed on top or bottom position and 20% - 25% of the chart's width vertically, when placed on left or right position of the chart. You can change this default legend size by using the [width](#) and [height](#) property of the `legendSettings`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' legendShape='Circle'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver' legendShape='Circle'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' legendShape='Circle'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public legendSettings?: Object;
  ngOnInit(): void {
    this.chartData = [
      { country: "USA", gold: 50, silver: 70, bronze: 45 },
      { country: "China", gold: 40, silver: 60, bronze: 55 },
      { country: "Japan", gold: 70, silver: 60, bronze: 50 },
      { country: "Australia", gold: 60, silver: 56, bronze: 40 },
      { country: "France", gold: 50, silver: 45, bronze: 35 },
      { country: "Germany", gold: 40, silver: 30, bronze: 22 },
      { country: "Italy", gold: 40, silver: 35, bronze: 37 },
      { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
    ];
    this.primaryXAxis = {
      valueType: 'Category',
      title: 'Countries'
    };
    this.primaryYAxis = {
      minimum: 0, maximum: 80,
      interval: 20, title: 'Medals',
    };
  }
}

```

```

        this.legendSettings = { visible: true, height: '100', width: '500'
    };
    this.title = 'Olympic Medals';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Legend Item Size

You can customize the size of the legend items by using the [shapeHeight](#) and [shapeWidth](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' legendShape='Rectangle'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver' legendShape='Rectangle'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' legendShape='Rectangle'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public legendSettings?: Object;
  ngOnInit(): void {
    this.chartData = [
      { country: "USA", gold: 50, silver: 70, bronze: 45 },

```

```

        { country: "China", gold: 40, silver: 60, bronze: 55 },
        { country: "Japan", gold: 70, silver: 60, bronze: 50 },
        { country: "Australia", gold: 60, silver: 56, bronze: 40 },
        { country: "France", gold: 50, silver: 45, bronze: 35 },
        { country: "Germany", gold: 40, silver: 30, bronze: 22 },
        { country: "Italy", gold: 40, silver: 35, bronze: 37 },
        { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
    ];
    this.primaryXAxis = {
        valueType: 'Category',
        title: 'Countries'
    };
    this.primaryYAxis = {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals',
    };
    this.legendSettings = { visible: true, shapeHeight: 15, shapeWidth:
15 };
    this.title = 'Olympic Medals';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Paging for Legend

Paging will be enabled by default, when the legend items exceeds the legend bounds. You can view each legend items by navigating between the pages using navigation buttons.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, LegendService } from
'@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ CategoryService, LineSeriesService, LegendService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[legendSettings]='legendSettings' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='December 2007' width=2 [marker]='marker'></e-series>

```

```

        <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y1' name='December 2008' width=2 [marker]='marker1'></e-series>
        <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y2' name='December 2009' width=2 [marker]='marker2'></e-series>
        <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y3' name='December 2010' width=2 [marker]='marker3'></e-series>
    </e-series-collection>
</ejs-chart>`
    ))
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    public marker?: Object;
    public marker1?: Object;
    public marker2?: Object;
    public marker3?: Object;
    public legendSettings?: Object;
    ngOnInit(): void {
        this.chartData = [
            { x: 'WW', y: 12, y1: 22, y2: 38.3, y3: 50 },
            { x: 'EU', y: 9.9, y1: 26, y2: 45.2, y3: 63.6 },
            { x: 'APAC', y: 4.4, y1: 9.3, y2: 18.2, y3: 20.9 },
            { x: 'LATAM', y: 6.4, y1: 28, y2: 46.7, y3: 65.1 },
            { x: 'MEA', y: 30, y1: 45.7, y2: 61.5, y3: 73 },
            { x: 'NA', y: 25.3, y1: 35.9, y2: 64, y3: 81.4 }
        ];
        this.primaryXAxis = {
            title: 'Countries',
            valueType: 'Category', interval: 1,
            labelIntersectAction : 'Rotate45'
        };
        this.primaryYAxis = {
            title: 'Penetration (%)',
            rangePadding: 'None', labelFormat: '{value}%',
            minimum: 0, maximum: 90
        };
        this.marker = { visible: true, width: 10, height: 10, shape:
'Diamond' };
        this.marker1 = { visible: true, width: 10, height: 10, shape:
'Pentagon' };
        this.marker2 = { visible: true, width: 10, height: 10, shape:
'Triangle' };
        this.marker3 = { visible: true, width: 10, height: 10, shape:
'Circle' };
        this.title = 'FB Penetration of Internet Audience';
        this.legendSettings = {
            padding: 10, shapePadding: 10,
            visible: true, border: {
                width: 2, color: 'grey'
            },
            width: '200'
        };
    }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Legend Text Wrap

When the legend text exceeds the container, the text can be wrapped by using [textWrap](#) Property. End user can also wrap the legend text based on the [maximumLabelWidth](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold Medals'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver Medals'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze Medals'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public legendSettings?: Object;
  ngOnInit(): void {
    this.chartData = [
      { country: "USA", gold: 50, silver: 70, bronze: 45 },
      { country: "China", gold: 40, silver: 60, bronze: 55 },
      { country: "Japan", gold: 70, silver: 60, bronze: 50 },
      { country: "Australia", gold: 60, silver: 56, bronze: 40 },
      { country: "France", gold: 50, silver: 45, bronze: 35 },
```



```

        { country: "Germany", gold: 40, silver: 30, bronze: 22 },
        { country: "Italy", gold: 40, silver: 35, bronze: 37 },
        { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
    ];
    this.primaryXAxis = {
        valueType: 'Category',
        title: 'Countries'
    };
    this.primaryYAxis = {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals',
    };
    this.legendSettings = { visible: true, position: 'Right',
textWrap: 'Wrap',          maximumLabelWidth: 50 };
    this.title = 'Olympic Medals';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Set the label color based on series color

You can set the legend label color based on series color by using chart's [loaded](#) event.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { ILoadedEventArgs } from '@syncfusion/ej2-angular-charts';
import { categoryData } from './datasource';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="charts"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings' (loaded)="onChartLoaded($event)" >
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver'></e-series>
    </e-series-collection>
`
})
export class AppComponent implements OnInit {
    primaryXAxis: any;
    primaryYAxis: any;
    legendSettings: any;
    title: string;
    chartData: any;

    ngOnInit(): void {
        this.primaryXAxis = {
            valueType: 'Category',
            title: 'Countries'
        };
        this.primaryYAxis = {
            minimum: 0, maximum: 80,
            interval: 20, title: 'Medals'
        };
        this.legendSettings = {
            visible: true, position: 'Right',
            textWrap: 'Wrap', maximumLabelWidth: 50
        };
        this.title = 'Olympic Medals';
        this.chartData = categoryData;
    }
}

```

```

        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' ></e-series>
    </e-series-collection>
</ejs-chart>`
    })
    // declare the series colors
    export class AppComponent implements OnInit {
        public color: string[] = ['#00BDAE', '#404041', '#357CD2'];
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public primaryYAxis?: Object;
        public legendSettings?: Object;
        ngOnInit(): void {
            this.chartData = categoryData;
            this.primaryXAxis = {
                valueType: 'Category',
                title: 'Countries'
            };
            this.primaryYAxis = {
                minimum: 0, maximum: 80,
                interval: 20, title: 'Medals',
            };
            this.legendSettings = { visible: true, position: 'Top' };
            this.title = 'Olympic Medals';
        }
        public onChartLoaded(args: ILoadedEventArgs): void {
            let legendTextCol: any =
document.querySelectorAll('[id*="charts_chart_legend_text_"]');
            for (let i = 0; i < legendTextCol.length; i++) {
                //set the color to legend label
                legendTextCol[i].setAttribute('fill', this.color[i]);
            }
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series Selection on Legend

By default, legend click enables you to collapse the series visibility. On other hand, if you need to select a series through legend click, disable the [toggleVisibility](#).

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'

```

```

import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings' selectionMode='Point'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' legendShape='Circle'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver' legendShape='Circle'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' legendShape='Circle'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public legendSettings?: Object;
  ngOnInit(): void {
    this.chartData = [
      { country: "USA", gold: 50, silver: 70, bronze: 45 },
      { country: "China", gold: 40, silver: 60, bronze: 55 },
      { country: "Japan", gold: 70, silver: 60, bronze: 50 },
      { country: "Australia", gold: 60, silver: 56, bronze: 40 },
      { country: "France", gold: 50, silver: 45, bronze: 35 },
      { country: "Germany", gold: 40, silver: 30, bronze: 22 },
      { country: "Italy", gold: 40, silver: 35, bronze: 37 },
      { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
    ];
    this.primaryXAxis = {
      valueType: 'Category',
      title: 'Countries'
    };
    this.primaryYAxis = {
      minimum: 0, maximum: 80,
      interval: 20, title: 'Medals',
    };
    this.legendSettings = { visible: true, toggleVisibility: false };
    this.title = 'Olympic Medals';
  }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Enable Animation

You can customize the animation while clicking legend by setting enableAnimation as true or false using enableAnimation property in chart.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings' [enableAnimation]='enableAnimation'>
  <e-series-collection>
    <e-series [dataSource]='chartData1' type='Column' xName='x'
yName='y' name='Gold' [marker]='marker'></e-series>
    <e-series [dataSource]='chartData2' type='Column' xName='x'
yName='y' name='Silver' [marker]='marker'></e-series>
    <e-series [dataSource]='chartData3' type='Column' xName='x'
yName='y' name='Bronze' [marker]='marker'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData1?: Object[];
  public chartData2?: Object[];
  public chartData3?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public legendSettings?: Object;
  public marker?: Object;
  public enableAnimation: boolean = true;
  ngOnInit(): void {
    this.chartData1 = [{ x: 'USA', y: 46 }, { x: 'GBR', y: 27 }, { x:
'CHN', y: 26 }];
```

```

    this.chartData2 = [{ x: 'USA', y: 37 }, { x: 'GBR', y: 23 }, { x:
'CHN', y: 18 }];
    this.chartData3 = [{ x: 'USA', y: 38 }, { x: 'GBR', y: 17 }, { x:
'CHN', y: 26 }];
    this.primaryXAxis = {
        valueType: 'Category', interval: 1, majorGridLines: { width: 0 }
    };
    this.primaryYAxis = {
        majorGridLines: { width: 0 },
        majorTickLines: { width: 0 }, lineStyle: { width: 0 },
        labelStyle: { color: 'transparent' }
    };
    this.legendSettings = { visible: true };
    this.marker = { dataLabel: { visible: true, position: 'Top', font:
{ fontWeight: '600', color: '#ffffff' } } };
    this.title = 'Olympic Medal Counts - RIO';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Collapsing Legend Item

By default, series name will be displayed as legend. To skip the legend for a particular series, you can give empty string to the series name.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' legendShape='Circle'></e-series>
    </e-series-collection>
  `
})

```

```

        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' legendShape='Circle'></e-series>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' legendShape='Circle'></e-series>
    </e-series-collection>
</ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public primaryYAxis?: Object;
        public legendSettings?: Object;
        ngOnInit(): void {
            this.chartData = [
                { country: "USA", gold: 50, silver: 70, bronze: 45 },
                { country: "China", gold: 40, silver: 60, bronze: 55 },
                { country: "Japan", gold: 70, silver: 60, bronze: 50 },
                { country: "Australia", gold: 60, silver: 56, bronze: 40 },
                { country: "France", gold: 50, silver: 45, bronze: 35 },
                { country: "Germany", gold: 40, silver: 30, bronze: 22 },
                { country: "Italy", gold: 40, silver: 35, bronze: 37 },
                { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
            ];
            this.primaryXAxis = {
                valueType: 'Category',
                title: 'Countries'
            };
            this.primaryYAxis = {
                minimum: 0, maximum: 80,
                interval: 20, title: 'Medals',
            };
            this.legendSettings = { visible: true, toggleVisibility: true };
            this.title = 'Olympic Medals';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Legend Title

You can set title for legend using `title` property in `legendSettings`. You can also customize the `fontStyle`, `size`, `fontWeight`, `color`, `textAlignment`, `fontFamily`, `opacity` and `textOverflow` of legend title. `titlePosition` is used to set the legend position in `Top`, `Left` and `Right` position. `maximumTitleWidth` is used to set the width of the legend title. By default, it will be `100px`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'

```

```

import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings' selectionMode='Point'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' legendShape='Circle'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver' legendShape='Circle'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' legendShape='Circle'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public legendSettings?: Object;
  ngOnInit(): void {
    this.chartData = [
      { country: "USA", gold: 50, silver: 70, bronze: 45 },
      { country: "China", gold: 40, silver: 60, bronze: 55 },
      { country: "Japan", gold: 70, silver: 60, bronze: 50 },
      { country: "Australia", gold: 60, silver: 56, bronze: 40 },
      { country: "France", gold: 50, silver: 45, bronze: 35 },
      { country: "Germany", gold: 40, silver: 30, bronze: 22 },
      { country: "Italy", gold: 40, silver: 35, bronze: 37 },
      { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
    ];
    this.primaryXAxis = {
      valueType: 'Category',
    };
    this.primaryYAxis = {
      minimum: 0, maximum: 80,
      interval: 20, title: 'Medals',
    };
    this.legendSettings = { visible: true, title: 'Countries' };
    this.title = 'Olympic Medals';
  }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Arrow Page Navigation

By default, the page number will be enabled while legend paging. Now, you can disable that page number and also you can get left and right arrows for page navigation. You have to set `false` value to `enablePages` to get this support.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' [marker]='marker' type='Line'
name='December 2007' xName='x' yName='y'></e-series>
      <e-series [dataSource]='chartData' [marker]='marker' type='Line'
name='December 2008' xName='x' yName='y1'></e-series>
      <e-series [dataSource]='chartData' [marker]='marker' type='Line'
name='December 2009' xName='x' yName='y2'></e-series>
      <e-series [dataSource]='chartData' [marker]='marker' type='Line'
name='December 2010' xName='x' yName='y3'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public legendSettings?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.chartData = [
      { x: 'WW', y: 12, y1: 22, y2: 38.3, y3: 50 },
      { x: 'EU', y: 9.9, y1: 26, y2: 45.2, y3: 63.6 },
      { x: 'APAC', y: 4.4, y1: 9.3, y2: 18.2, y3: 20.9 },
      { x: 'LATAM', y: 6.4, y1: 28, y2: 46.7, y3: 65.1 },
```



```

    { x: 'MEA', y: 30, y1: 45.7, y2: 61.5, y3: 73 },
    { x: 'NA', y: 25.3, y1: 35.9, y2: 64, y3: 81.4 }
  ];
  this.primaryXAxis = {
    valueType: 'Category',
  };
  this.primaryYAxis = {
    minimum: 0, maximum: 80,
    interval: 20, title: 'Medals',
  };
  this.legendSettings = { width: '180', enablePages: false };
  this.title = 'Olympic Medals';
  this.marker = {
    visible: true,
    width: 10, height: 10,
    shape: 'Diamond'
  }
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Legend Item Padding

The [itemPadding](#) property can be used to adjust the space between the legend items.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' [marker]='marker' type='Line'
name='December 2007' xName='x' yName='y'></e-series>
      <e-series [dataSource]='chartData' [marker]='marker' type='Line'
name='December 2008' xName='x' yName='y1'></e-series>
    </e-series-collection>
  `
})
export class AppComponent implements OnInit {
  primaryXAxis: string[] = ['MEA', 'NA'];
  primaryYAxis: number[] = [0, 20, 40, 60, 80];
  legendSettings: LegendSettings = { width: '180', enablePages: false };
  title: string = 'Olympic Medals';
  chartData: ChartData = [
    { x: 'MEA', y: 30, y1: 45.7, y2: 61.5, y3: 73 },
    { x: 'NA', y: 25.3, y1: 35.9, y2: 64, y3: 81.4 }
  ];
  marker: Marker = { visible: true, width: 10, height: 10, shape: 'Diamond' };
  ngOnInit(): void {
  }
}

```

```

        <e-series [dataSource]='chartData' [marker]='marker' type='Line'
name='December 2009' xName='x' yName='y2'></e-series>
        <e-series [dataSource]='chartData' [marker]='marker' type='Line'
name='December 2010' xName='x' yName='y3'></e-series>
    </e-series-collection>
</ejs-chart>`
}))
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    public legendSettings?: Object;
    public marker?: Object;
    ngOnInit(): void {
        this.chartData = [
            { x: 'WW', y: 12, y1: 22, y2: 38.3, y3: 50 },
            { x: 'EU', y: 9.9, y1: 26, y2: 45.2, y3: 63.6 },
            { x: 'APAC', y: 4.4, y1: 9.3, y2: 18.2, y3: 20.9 },
            { x: 'LATAM', y: 6.4, y1: 28, y2: 46.7, y3: 65.1 },
            { x: 'MEA', y: 30, y1: 45.7, y2: 61.5, y3: 73 },
            { x: 'NA', y: 25.3, y1: 35.9, y2: 64, y3: 81.4 }
        ];
        this.primaryXAxis = {
            valueType: 'Category',
        };
        this.primaryYAxis = {
            minimum: 0, maximum: 80,
            interval: 20, title: 'Medals';
        };
        this.legendSettings = { enablePages: false, position: "Bottom",
            itemPadding: 30 };
        this.title = 'Olympic Medals';
        this.marker = {
            visible: true,
            width: 10, height: 10,
            shape: 'Diamond'
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note: To use legend feature, we need to inject `LegendService` into the `@NgModule.Providers`.

See Also

- [Customize each shape in legend](#)

Tooltip in Angular Chart component

<!-- markdownlint-disable MD036 -->

Chart will display details about the points through tooltip, when the mouse is moved over the point.

Default tooltip

By default, tooltip is not visible. You can enable the tooltip by setting [enable](#) property to [Link to the Video](#) and by injecting `TooltipService` into the `NgModule.providers`.

To know about tooltip, you can check on this video:

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, StepLineSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, TooltipService, CategoryService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { toolData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, StepLineSeriesService, LegendService,
    TooltipService, CategoryService, ColumnSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[tooltip]='tooltip'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y' width=2 name='China' [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
  public tooltip?: Object;
  ngOnInit(): void {
    this.chartData = toolData;
    this.primaryXAxis = {
      valueType: 'DateTime',
    };
    this.tooltip = { enable: true };
    this.marker = { visible: true, width: 10, height: 10 };
    this.title = 'Unemployment Rates 1975-2010';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

<!-- markdownlint-disable MD013 -->

Fixed tooltip

By default, tooltip track the mouse movement, but you can set a fixed position for the tooltip by using the [location](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, StepLineSeriesService, ColumnSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { LegendService, TooltipService, CategoryService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { toolData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, StepLineSeriesService, LegendService,
    TooltipService, CategoryService, ColumnSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[tooltip]='tooltip'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y' width=2 name='China' [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
  public tooltip?: Object;
  ngOnInit(): void {
    this.chartData = toolData;
    this.primaryXAxis = {
      valueType: 'DateTime',
    };
    this.tooltip = { enable: true, location: { x: 120, y: 20 } };
    this.marker = { visible: true, width: 10, height: 10 };
    this.title = 'Unemployment Rates 1975-2010';
```

```
}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Format the tooltip

```
<!-- markdownlint-disable MD013 -->
```

By default, tooltip shows information of x and y value in points. In addition to that, you can show more information in tooltip. For example the format `${series.name} ${point.x}` shows series name and point x value.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, StepLineSeriesService, ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, TooltipService, CategoryService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { toolData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, StepLineSeriesService, LegendService,
    TooltipService, CategoryService, ColumnSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[tooltip]='tooltip'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y' width=2 name='China' [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  public tooltip?: Object;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = toolData;
    this.primaryXAxis = {
```

```

        valueType: 'DateTime'
    };
    this.tooltip = { enable: true, header: 'Unemployment', format:
'<b>${point.x} : ${point.y}</b>' };
    this.marker = { visible: true, width: 10, height: 10 };
    this.title = 'Unemployment Rates 1975-2010';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD013 -->

Individual series format

<!-- markdownlint-disable MD013 -->

You can format the each series tooltip separately using series [tooltipFormat](#) property.

Note: If series [tooltipFormat](#) is given, it shows the tooltip for that series in that format, or else it will take tooltip format.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, StepLineSeriesService, ColumnSeriesService } from
'@syncfusion/ej2-angular-charts'
import { LegendService, TooltipService, CategoryService } from
'@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { toolData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, StepLineSeriesService, LegendService,
  TooltipService, CategoryService, ColumnSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[title]='title' [tooltip]='tooltip'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y' width=2 name='China' [marker]='marker'
[tooltipFormat]="tooltipFormat"></e-series>
      <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y1' width=2 name='China' [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})

```

```
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public marker?: Object;
    public tooltip?: Object;
    public tooltipFormat?: string;
    ngOnInit(): void {
        this.chartData = toolData;
        this.primaryXAxis = {
            valueType: 'DateTime'
        };
        this.tooltip = { enable: true, header: 'Unemployment', format:
'<b>${point.x} : ${point.y}</b>' };
        this.marker = { visible: true, width: 10, height: 10 };
        this.title = 'Unemployment Rates 1975-2010';
        this.tooltipFormat = '${point.x}';
    }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

<!-- markdownlint-disable MD013 -->

Tooltip template

Any HTML elements can be displayed in the tooltip by using the [template](#) property of the tooltip. You can use the `${x}` and `${y}` as place holders in the HTML element to display the x and y values of the corresponding data point.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, StepLineSeriesService, ColumnSeriesService } from
'@syncfusion/ej2-angular-charts'
import { LegendService, TooltipService, CategoryService } from
'@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { toolData } from './datasource';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ DateTimeService, StepLineSeriesService, LegendService,
        TooltipService, CategoryService, ColumnSeriesService ],
    standalone: true,
    selector: 'app-container',
```

```

    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[tooltip]='tooltip'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y' width=2 name='China' [marker]='marker'></e-series>
    </e-series-collection>
</ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    public marker?: Object;
    public tooltip?: Object;
    ngOnInit(): void {
      this.chartData = toolData;
      this.primaryXAxis = {
        valueType: 'DateTime'
      };
      this.tooltip = { enable: true, template: '#Unemployment' };
      this.marker = { visible: true, width: 10, height: 10 };
      this.title = 'Unemployment Rates 1975-2010';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Tooltip mapping name

By default, tooltip shows information of x and y value in points. You can show more information from data source in tooltip by using the [tooltipMappingName](#) property of the tooltip. You can use the `${point.tooltip}` as place holders to display the specified tooltip content.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, StepLineSeriesService, ColumnSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { LegendService, TooltipService, CategoryService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { toolData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],

```



```

providers: [ DateTimeService, StepLineSeriesService, LegendService,
TooltipService, CategoryService, ColumnSeriesService],
standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[title]='title' [tooltip]='tooltip'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' width=2 tooltipMappingName='country'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public tooltip?: Object;
  ngOnInit(): void {
    this.chartData = [
      { x: 'Germany', y: 72, country: 'GER: 72' },
      { x: 'Russia', y: 103.1, country: 'RUS: 103.1' },
      { x: 'Brazil', y: 139.1, country: 'BRZ: 139.1' },
      { x: 'India', y: 462.1, country: 'IND: 462.1' },
      { x: 'China', y: 721.4, country: 'CHN: 721.4' },
      { x: 'United States Of America', y: 286.9, country:
'USA: 286.9' },
      { x: 'Great Britain', y: 115.1, country: 'GBR: 115.1' },
      { x: 'Nigeria', y: 97.2, country: 'NGR: 97.2' },
    ];
    this.primaryXAxis = {
      valueType: 'Category',
    };
    this.tooltip = {
      enable: true, format: '${point.tooltip}'
    };
    this.title = 'Internet Users in Million - 2016';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customize the appearance of tooltip

The [fill](#) and [border](#) properties are used to customize the background color and border of the tooltip respectively. The [textStyle](#) property in the tooltip is used to customize the font of the tooltip text. The [highlightColor](#) property is used to customize the point color while hovering for tooltip.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'

```

```

import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, StepLineSeriesService, ColumnSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { LegendService, TooltipService, CategoryService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { toolData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, StepLineSeriesService, LegendService,
    TooltipService, CategoryService, ColumnSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" highlightColor="red"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
    [tooltip]='tooltip'>
      <e-series-collection>
        <e-series [dataSource]='chartData' type='StepLine' xName='x'
        yName='y' width=2 name='China' [marker]='marker'></e-series>
      </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
  public tooltip?: Object;
  ngOnInit(): void {
    this.chartData = toolData;
    this.primaryXAxis = {
      valueType: 'DateTime'
    };
    this.tooltip = {
      enable: true,
      format: '${series.name} ${point.x} : ${point.y}',
      fill: '#7bb4eb',
      border: {
        width: 2,
        color: 'grey'
      }
    };
    this.marker = { visible: true, width: 10, height: 10 };
    this.title = 'Unemployment Rates 1975-2010';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See also

- [Format the tooltip value](#)
- [Create a table in tooltip](#)

Zooming in Angular Chart component

Enable zooming

Chart can be zoomed in three ways.

- Selection - By setting [enableSelectionZooming](#) property to true in `zoomSettings`, you can zoom the chart by using the rubber band selection.
- Mousewheel - By setting [enableMouseWheelZooming](#) property to true in `zoomSettings`, you can zoom in and zoom out the chart by scrolling the mouse wheel.
- Pinch - By setting [enablePinchZooming](#) property to true in [Link to the Video](#), you can zoom the chart through pinch gesture in touch enabled devices.

Pinch zooming is supported only in browsers that support multi-touch gestures. Currently IE11, Chrome and Opera browsers support multi-touch in desktop devices.

To know about Zooming and Panning, you can check on this video:

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, AreaSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, ZoomService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { series1 } from '../datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, AreaSeriesService, LegendService,
    ZoomService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
    [zoomSettings]='zoom' [legendSettings]='legend'>
      <e-series-collection>
        <e-series [dataSource]='chartData' type='Area' xName='x'
          yName='y' name='Product X' [border]='border' [animation]='animation'
          opacity=0.3></e-series>
      </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
```

```

public chartData?: Object[];
public title?: string;
public primaryYAxis?: Object;
public border?: Object;
public zoom?: Object;
public animation?: Object;
public legend?: Object;
ngOnInit(): void {
    this.chartData = series1;
    this.primaryXAxis = {
        valueType: 'DateTime',
        labelFormat: 'yMMM',
    };
    this.zoom = {
        enableMouseWheelZooming: true,
        enablePinchZooming: true,
        enableSelectionZooming: true
    };
    this.animation = { enable: false };
    this.legend = { visible: false };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note: To use zooming feature, we need to inject `ZoomService` into the `NgModule.providers`.

After zooming the chart, a zooming toolbar will appear with `zoom`, `zoomin`, `zoomout`, `pan` and `reset` buttons.

Selecting the Pan option will allow to pan the chart and selecting the Reset option will reset the zoomed chart.

Modes

The `mode` property in `zoomSettings` specifies whether the chart is allowed to scale along the horizontal axis or vertical axis. The default value of the mode is XY (both axis).

There are three types of mode.

- X - Allows us to zoom the chart horizontally.
- Y - Allows us to zoom the chart vertically.
- XY - Allows us to zoom the chart both vertically and horizontally.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChartModule } from '@syncfusion/ej2-angular-charts';

```

```

import { DateTimeService, AreaSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, ZoomService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { series1 } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, AreaSeriesService, LegendService, ZoomService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[zoomSettings]='zoom' [legendSettings]='legend'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Area' xName='x'
yName='y' name='Product X' [border]='border' [animation]='animation'
opacity=0.3></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public primaryYAxis?: Object;
  public zoom?: Object;
  public animation?: Object;
  public legend?: Object;
  title: any;
  border: any;
  ngOnInit(): void {
    this.chartData = series1;
    this.primaryXAxis = {
      valueType: 'DateTime',
      labelFormat: 'yMMM',
    };
    this.zoom = {
      enableSelectionZooming: true,
      mode: 'X'
    };
    this.animation = { enable: false };
    this.legend = { visible: false };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Toolbar

By default, zoomin, zoomout, pan and reset buttons will be displayed for zoomed chart. You can customize to show the desired options in the toolbar using the [toolbarItems](#) property. Also using the [showToolbar](#) property, you can show toolkit for zooming and panning the chart during initial rendering itself.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, AreaSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, ZoomService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { series1 } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, AreaSeriesService, LegendService,
    ZoomService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[zoomSettings]='zoom' [legendSettings]='legend'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Area' xName='x'
yName='y' name='Product X' [border]='border' [animation]='animation'
opacity=0.3></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public primaryYAxis?: Object;
  public zoom?: Object;
  public animation?: Object;
  public legend?: Object;
  title: any;
  border: any;
  ngOnInit(): void {
    this.chartData = series1;
    this.primaryXAxis = {
      valueType: 'DateTime',
      labelFormat: 'yMMM',
    };
    this.zoom = {
      enableSelectionZooming: true,
      toolbarItems: ['Zoom', 'Pan', 'Reset'],
      showToolbar: true
    };
    this.animation = { enable: false };
    this.legend = { visible: false };
  }
}
```

```
}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Enable pan

Using [enablePan](#) property you can able to pan the zoomed chart without help of toolbar items.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, AreaSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, ZoomService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { series1 } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, AreaSeriesService, LegendService,
    ZoomService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[zoomSettings]='zoom' [legendSettings]='legend'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Area' xName='x'
yName='y' name='Product X' [border]='border' [animation]='animation'
opacity=0.3></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public primaryYAxis?: Object;
  public zoom?: Object;
  public animation?: Object;
  public legend?: Object;
  title: any;
  border: any;
  ngOnInit(): void {
    this.chartData = series1;
    this.primaryXAxis = {
      valueType: 'DateTime',
      labelFormat: 'yMMM',
      zoomFactor: 0.2, zoomPosition: 0.6
```

```

    };
    this.zoom = {
        enableSelectionZooming: true,
        enablePan: true
    };
    this.animation = { enable: false };
    this.legend = { visible: false };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Enable scrollbar

Using the [enableScrollbar](#) property, you can add a scrollbar to a zoomed chart. This scrollbar allows you to zoom or pan the chart. The appearance of the scrollbar can be customized using properties in [scrollbarSettings](#). For example, you can use [trackColor](#) and [trackRadius](#) properties to customize the track of the scrollbar, and [scrollbarRadius](#) and [scrollbarColor](#) properties to customize the scroller. The ability to zoom through the scrollbar can be enabled or disabled using the [enableZoom](#) property in [scrollbarSettings](#). Additionally, you can change the color of the grip and height of the scrollbar using the [gripColor](#) and [height](#) properties.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChartModule } from '@syncfusion/ej2-angular-charts';
import { DateTimeService, AreaSeriesService } from '@syncfusion/ej2-angular-charts';
import { LegendService, ZoomService, ScrollBarService } from '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
import { series1 } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, AreaSeriesService, LegendService, ZoomService, ScrollBarService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[zoomSettings]='zoom' [legendSettings]='legend'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Area' xName='x'
yName='y' name='Product X' [border]='border' [animation]='animation'
opacity=0.3></e-series>
    </e-series-collection>
  </ejs-chart>`
})

```



```

    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public primaryYAxis?: Object;
        public zoom?: Object;
        public animation?: Object;
        public legend?: Object;
        title: any;
        border: any;
        ngOnInit(): void {
            this.chartData = series1;
            this.primaryXAxis = {
                valueType: 'DateTime',
                labelFormat: 'yMMM',
                zoomFactor: 0.2, zoomPosition: 0.6,
                scrollbarSettings: {
                    enable: true,
                    enableZoom: false,
                    height: 14,
                    trackRadius: 8,
                    scrollbarRadius: 8,
                    gripColor: 'transparent',
                    trackColor: 'yellow',
                    scrollbarColor: 'red'
                }
            };
            this.zoom = {
                enableSelectionZooming: true,
                enableScrollbar: true
            };
            this.animation = { enable: false };
            this.legend = { visible: false };
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Auto interval on zooming

By using [enableAutoIntervalOnZooming](#) property, the axis interval will get calculated automatically with respect to the zoomed range.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, AreaSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, ZoomService } from '@syncfusion/ej2-angular-charts'

```

```

import { Component, OnInit } from '@angular/core';
import { series1 } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, AreaSeriesService, LegendService,
    ZoomService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[zoomSettings]='zoom' [legendSettings]='legend'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Area' xName='x'
yName='y' name='Product X' [border]='border' [animation]='animation'
opacity=0.3></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public primaryYAxis?: Object;
  public zoom?: Object;
  public animation?: Object;
  public legend?: Object;
  title: any;
  border: any;
  ngOnInit(): void {
    this.chartData = series1;
    this.primaryXAxis = {
      valueType: 'DateTime',
      labelFormat: 'yMMM',
      enableAutoIntervalOnZooming: true
    };
    this.zoom = {
      enableSelectionZooming: true,
    };
    this.animation = { enable: false };
    this.legend = { visible: false };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

Data editing in Angular Chart component

Enable Data Editing

We can use the data editing through inject the `DataEditingService` module. It provides drag and drop support to the rendered points. Now, we can change the location or value of the point based on its `y` value. To enable the data editing, set the `enable` property to true in the drag settings of the series. Also, we can set color using `fill` property and set the data editing minimum and maximum range using `minY` and `maxY` properties.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { LineSeriesService, ColumnSeriesService, CategoryService,
DataEditingService, TooltipService } from '@syncfusion/ej2-angular-charts'
import { LegendService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ LegendService, LineSeriesService, ColumnSeriesService,
CategoryService, DataEditingService, TooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis' [title]='title'
[chartArea]="chartArea" [tooltip]="tooltip">
  <e-series-collection>
    <e-series [dataSource]='columnData' type='Column' xName='x'
yName='y' width="2" [marker]="marker" [dragSettings]="dragSettings"></e-
series>
    <e-series [dataSource]='lineData' type='Line' xName='x'
yName='y' width="2" [marker]="marker" [dragSettings]="dragSettings"></e-
series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public columnData?: Object[];
  public lineData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public chartArea?: Object;
  public marker?: Object;
  public dragSettings?: Object;
  public tooltip?: Object;
  ngOnInit(): void {
    this.columnData = [
      { x: '2005', y: 21 }, { x: '2006', y: 60 },
      { x: '2007', y: 45 }, { x: '2008', y: 50 },
      { x: '2009', y: 74 }, { x: '2010', y: 65 },
      { x: '2011', y: 85 }
    ];
  }
}
```

```

    this.lineData = [
      { x: '2005', y: 21 }, { x: '2006', y: 22 },
      { x: '2007', y: 36 }, { x: '2008', y: 34 },
      { x: '2009', y: 54 }, { x: '2010', y: 55 },
      { x: '2011', y: 60 }
    ];
    this.primaryXAxis = {
      valueType: 'Category',
      minimum: -0.5,
      maximum: 6.5,
      labelPlacement: 'OnTicks',
      majorGridLines: { width: 0 },
    };
    this.primaryYAxis = {
      rangePadding: 'None',
      minimum: 0,
      title: 'Sales',
      labelFormat: '{value}%',
      maximum: 100,
      interval: 20,
      lineStyle: { width: 0 },
      majorTickLines: { width: 0 },
      minorTickLines: { width: 0 }
    };
    this.chartArea = {
      border: {
        width: 0
      }
    };
    this.title = 'Inflation - Consumer Price';
    this.marker = {
      visible: true,
      width: 10,
      height: 10
    };
    this.dragSettings = {
      enable: true
    };
    this.tooltip = {
      enable: true
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Cross hair and track ball in Angular Chart component

Crosshair has a vertical and horizontal line to view the value of the axis at mouse or touch position.

Crosshair lines can be enabled by using [enable](#) property in the `crosshair`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, CrosshairService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { ChartData } from './chartdata';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, LegendService,
    CrosshairService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legend' [crosshair]='crosshair'>
    <e-series-collection>
      <e-series [dataSource]='series1' type='Line' xName='x' yName='y'
name='Temperature'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public series1?: Object[];
  public crosshair?: Object;
  public title?: string;
  public primaryYAxis?: Object;
  public legend?: Object;
  ngOnInit(): void {
    this.primaryXAxis = {
      valueType: 'DateTime',
      labelFormat: 'yMMM'
    };
    this.crosshair = { enable: true };
    this.series1 = ChartData.prototype.getCrosshairData().series1;
    this.legend = { visible: true };
    this.title = 'Weather Condition';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Tooltip for axis

Tooltip label for an axis can be enabled by using [enable](#)

property of `crosshairTooltip` in the corresponding axis.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, CrosshairService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { ChartData } from './chartdata';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, LegendService, CrosshairService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legend' [crosshair]='crosshair'>
    <e-series-collection>
      <e-series [dataSource]='series1' type='Line' xName='x' yName='y'
name='Temperature'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public series1?: Object[];
  public majorGridLines?: Object;
  public crosshair?: Object;
  public title?: string;
  public primaryYAxis?: Object;
  public legend?: Object;
  ngOnInit(): void {
    this.primaryXAxis = {
      valueType: 'DateTime',
      crosshairTooltip: { enable: true },
      labelFormat: 'yMMM'
    };
    this.primaryYAxis = {
      crosshairTooltip: { enable: true }
    };
    this.crosshair = { enable: true };
    this.series1 = ChartData.prototype.getCrosshairData().series1;
    this.legend = { visible: true };
    this.title = 'Weather Condition';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customization

The [fill](#) and [textStyle](#) property of the `crosshairTooltip` is used to customize the background color and font style of the crosshair label respectively. Color and width of the crosshair line can be customized by using the [line](#) property in the crosshair.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, CrosshairService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { ChartData } from './chartdata';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, LegendService,
    CrosshairService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
    [legendSettings]='legend' [crosshair]='crosshair'>
    <e-series-collection>
      <e-series [dataSource]='series1' type='Line' xName='x' yName='y'
        name='Temperature'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public series1?: Object[];
  public crosshair?: Object;
  public title?: string;
  public primaryYAxis?: Object;
  public legend?: Object;
  ngOnInit(): void {
    this.primaryXAxis = {
      valueType: 'DateTime',
      crosshairTooltip: { enable: true, fill: 'green' },
      labelFormat: 'yMMM'
    };
    this.primaryYAxis = {
      crosshairTooltip: { enable: true, fill: 'green' }
    };
  }
}
```

```

        this.crosshair = { enable: true, line: {width: 2, color: 'green'},
fill: 'green' };
        this.series1 = ChartData.prototype.getCrosshairData().series1;
        this.series1 = ChartData.prototype.getCrosshairData().series2;
        this.legend = { visible: true};
        this.title = 'Weather Condition';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note: To use crosshair feature, we need to inject `CrosshairService` into the `NgModule.providers`.

Trackball

Trackball is used to track a data point closest to the mouse or touch position. Trackball marker indicates the closest point and trackball tooltip displays the information about the point. To use trackball feature, we need to inject `CrosshairService` and `TooltipService` into the `NgModule.providers`.

Trackball can be enabled by setting the `enable` property of the crosshair to true and `shared` property in [Link to the Video](#) to true in chart.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, DataLabelService, TooltipService, CrosshairService }
from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, LegendService,
DataLabelService,
TooltipService, CrosshairService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[crosshair]='crosshair' [tooltip]='tooltip'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='John' width=2 [marker]='marker'></e-series>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y1' name='Andrew' width=2 [marker]='marker'></e-series>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y2' name='Thomas' width=2 [marker]='marker'></e-series>
    </e-series-collection>
  `
})
export class AppComponent implements OnInit {
  ngOnInit(): void {
  }
}

```



```

        <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y3' name='Mark' width=2 [marker]='marker'></e-series>
        <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y4' name='William' width=2 [marker]='marker'></e-series>
    </e-series-collection>
</ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public primaryYAxis?: Object;
        public chartData?: Object[];
        public crosshair?: Object;
        public title?: string;
        public tooltip?: Object;
        public marker?: Object;
        ngOnInit(): void {
            this.chartData = [
85         { x: new Date(2000, 2, 11), y: 15, y1: 39, y2: 60, y3: 75, y4:
83         },
            { x: new Date(2000, 9, 14), y: 20, y1: 30, y2: 55, y3: 75, y4:
85         },
            { x: new Date(2001, 2, 11), y: 25, y1: 28, y2: 48, y3: 68, y4:
87         },
            { x: new Date(2001, 9, 16), y: 21, y1: 35, y2: 57, y3: 75, y4:
            },
            { x: new Date(2002, 2, 7), y: 13, y1: 39, y2: 62, y3: 71, y4: 82
            },
            { x: new Date(2002, 9, 7), y: 18, y1: 41, y2: 64, y3: 69, y4: 74
            },
73         { x: new Date(2003, 2, 11), y: 24, y1: 45, y2: 57, y3: 81, y4:
75         },
            { x: new Date(2003, 9, 14), y: 23, y1: 48, y2: 53, y3: 84, y4:
            },
            { x: new Date(2004, 2, 6), y: 19, y1: 54, y2: 63, y3: 85, y4: 73
            },
            { x: new Date(2004, 9, 6), y: 31, y1: 55, y2: 50, y3: 87, y4: 60
            },
            { x: new Date(2005, 2, 11), y: 39, y1: 57, y2: 66, y3: 75, y4:
48         },
            { x: new Date(2005, 9, 11), y: 50, y1: 60, y2: 65, y3: 70, y4:
55         },
            { x: new Date(2006, 2, 11), y: 24, y1: 60, y2: 79, y3: 85, y4:
40         }
            ];
            this.primaryXAxis = {
                title: 'Years',
                minimum: new Date(2000, 1, 1), maximum: new Date(2006, 2, 11),
                intervalType: 'Years',
                valueType: 'DateTime',
            };
            this.tooltip = { enable: true, shared: true, format: '${series.name}
: ${point.x} : ${point.y}' };
            this.crosshair = { enable: true, lineType: 'Vertical' };
            this.marker = { visible: true };
            this.title = 'Average Sales per Person';
        }
    }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

To know about Crosshair and Trackball, you can check on this video:

Synchronized Charts in Angular Chart component

Tooltip synchronization

The tooltip can be synchronized across multiple charts using the [showTooltip](#) and [hideTooltip](#) methods. When we hover over a data point in one chart, we call the `showTooltip` method for the other charts to display related information in other connected charts simultaneously.

In the `showTooltip` method, specify the following parameters programmatically to enable tooltip for a particular chart:

- `x` - Data point x-value or x-coordinate value.
- `y` - Data point y-value or y-coordinate value.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, AreaSeriesService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { TooltipService } from '@syncfusion/ej2-angular-charts'
import { Component, ViewChild, OnInit } from '@angular/core';
import { IMouseEventArgs, ChartComponent } from '@syncfusion/ej2-angular-charts';
import { synchronizedData } from './datasource';
import { Browser } from '@syncfusion/ej2-base';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, AreaSeriesService, LineSeriesService,
    TooltipService ],
  standalone: true,
  selector: 'app-container',
  template: `<div class="control-section">
    <div class="row">
      <div class="col" >
        <ejs-chart #chart1 style='display:block;' id="container1"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis1'
          [title]='title1' [titleStyle]="titleStyle"
[tooltip]="tooltip1"
          (chartMouseLeave)= 'chart1MouseLeave($event)'
          (chartMouseMove)= 'chart1MouseMove($event)'
          (chartMouseUp)= 'chart1MouseUp($event)'>
```

```

        <e-series-collection>
            <e-series [dataSource]='chartData' type='Line'
xName='USD' yName='EUR' [width]="width">
            </e-series>
        </e-series-collection>
    </ejs-chart>
</div>
<div class="col" >
    <ejs-chart #chart2 style='display:block;' id="container2"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis2'
        [title]='title2' [titleStyle]="titleStyle"
[tooltip]="tooltip2"
        (chartMouseLeave)= 'chart2MouseLeave($event)'
(chartMouseMove)= 'chart2MouseMove($event)'
(chartMouseUp)= 'chart2MouseUp($event)'>
        <e-series-collection>
            <e-series [dataSource]='chartData' type='Area'
xName='USD' yName='INR' opacity=0.6
                [width]="width" [border]='border'>
            </e-series>
        </e-series-collection>
    </ejs-chart>
</div>
</div>`
}))
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public primaryYAxis1?: Object;
    public primaryYAxis2?: Object;
    public chartData?: Object[];
    public title1?: string;
    public title2?: string;
    public titleStyle?: Object;
    public tooltip1?: Object;
    public tooltip2?: Object;
    public border?: Object;
    public width?: number;
    @ViewChild('chart1')
    public chart1: ChartComponent;
    @ViewChild('chart2')
    public chart2: ChartComponent;
    public chart1MouseLeave(args: IMouseEventArgs): void {
        this.chart2.hideTooltip();
    };
    public chart1MouseMove(args: IMouseEventArgs): void {
        if ((!Browser.isDevice && !this.chart1.isTouch &&
!this.chart1.isChartDrag) || this.chart1.startMove) {
            this.chart2.startMove = this.chart1.startMove;
            this.chart2.showTooltip(args.x, args.y);
        }
    };
    public chart1MouseUp(args: IMouseEventArgs): void {
        if (Browser.isDevice && this.chart1.startMove) {
            this.chart2.hideTooltip();
        }
    };
};

```

```

public chart2MouseLeave(args: IMouseEventArgs): void {
    this.chart1.hideTooltip();
};
public chart2MouseMove(args: IMouseEventArgs): void {
    if ((!Browser.isDevice && !this.chart2.isTouch &&
!this.chart2.isChartDrag) || this.chart2.startMove) {
        this.chart2.startMove = this.chart1.startMove;
        this.chart1.showTooltip(args.x, args.y);
    }
};
public chart2MouseUp(args: IMouseEventArgs): void {
    if (Browser.isDevice && this.chart2.startMove) {
        this.chart1.hideTooltip();
    }
};
ngOnInit(): void {
    this.chartData = synchronizedData;
    this.primaryXAxis = {
        minimum: new Date(2023, 1, 18),
        maximum: new Date(2023, 7, 18),
        valueType: 'DateTime',
        labelFormat: 'MMM d',
        lineStyle: { width: 0 },
        majorGridLines: { width: 0 },
        edgeLabelPlacement: Browser.isDevice ? 'None' : 'Shift',
        labelRotation: Browser.isDevice ? -45 : 0,
        interval: Browser.isDevice ? 2 : 1
    };
    this.primaryYAxis1 = {
        labelFormat: 'n2',
        majorTickLines: { width: 0 },
        lineStyle: { width: 0 },
        minimum: 0.86,
        maximum: 0.96,
        interval: 0.025
    };
    this.primaryYAxis2 = {
        labelFormat: 'n1',
        majorTickLines: { width: 0 },
        lineStyle: { width: 0 },
        minimum: 79,
        maximum: 85,
        interval: 1.5
    };
    this.title1 = 'US to Euro';
    this.title2 = 'US to INR';
    this.tooltip1 = {
        enable: true, fadeOutDuration: Browser.isDevice ? 2500 : 1000,
        shared: true, header: '', format: '<b>€${point.y}</b><br>${point.x} 2023',
        enableMarker: false
    };
    this.tooltip2 = {
        enable: true, fadeOutDuration: Browser.isDevice ? 2500 : 1000,
        shared: true, header: '', format: '<b>₹${point.y}</b><br>${point.x} 2023',
        enableMarker: false
    };
    this.border = {

```

```

        width: 2
    };
    this.width = 2;
    this.titleStyle = {
        textAlignment: 'Near'
    };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Crosshair synchronization

The crosshair can be synchronized across multiple charts using the [showCrosshair](#) and [hideCrosshair](#) methods. When we hover over one chart, we call the `showCrosshair` method for the other charts to align with data points in other connected charts, simplifying data comparison and analysis.

In the `showCrosshair` method, specify the following parameters programmatically to enable crosshair for a particular chart:

- `x` - Specifies the x-value of the point or x-coordinate.
- `y` - Specifies the y-value of the point or y-coordinate.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, AreaSeriesService, SplineSeriesService } from '@syncfusion/ej2-angular-charts'
import { CrosshairService } from '@syncfusion/ej2-angular-charts'
import { Component, ViewChild, OnInit } from '@angular/core';
import { IMouseEventArgs, ChartComponent } from '@syncfusion/ej2-angular-charts';
import { synchronizedData } from './datasource';
import { Browser } from '@syncfusion/ej2-base';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, AreaSeriesService, SplineSeriesService, CrosshairService ],
  standalone: true,
  selector: 'app-container',
  template: `<div class="control-section">
<div class="row">
  <div class="col" >
    <ejs-chart #chart1 style='display:block;' id="container1"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis1'

```

```

        [title]='title1' [titleStyle]="titleStyle"
    [crosshair]='crosshair'
        (chartMouseLeave)= 'chart1MouseLeave($event)'
    (chartMouseMove)= 'chart1MouseMove($event)'
    (chartMouseUp)= 'chart1MouseUp($event)'>
        <e-series-collection>
            <e-series [dataSource]='chartData' type='Spline'
xName='USD' yName='EUR' [width]="width">
            </e-series>
        </e-series-collection>
    </ejs-chart>
</div>
<div class="col" >
    <ejs-chart #chart2 style='display:block;' id="container2"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis2'
        [title]='title2' [titleStyle]="titleStyle"
    [crosshair]='crosshair'
        (chartMouseLeave)= 'chart2MouseLeave($event)'
    (chartMouseMove)= 'chart2MouseMove($event)'
    (chartMouseUp)= 'chart2MouseUp($event)'>
        <e-series-collection>
            <e-series [dataSource]='chartData' type='Area'
xName='USD' yName='INR' opacity=0.6
                [width]="width" [border]='border'>
            </e-series>
        </e-series-collection>
    </ejs-chart>
</div>
</div>
</div>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public primaryYAxis1?: Object;
        public primaryYAxis2?: Object;
        public chartData?: Object[];
        public title1?: string;
        public title2?: string;
        public titleStyle?: Object;
        public border?: Object;
        public width?: number;
        public crosshair?: Object;
        @ViewChild('chart1')
        public chart1: ChartComponent;
        @ViewChild('chart2')
        public chart2: ChartComponent;
        public chart1MouseLeave(args: IMouseEventArgs): void {
            this.chart2.hideCrosshair();
        };
        public chart1MouseMove(args: IMouseEventArgs): void {
            if (!!Browser.isDevice && !this.chart1.isTouch &&
!this.chart1.isChartDrag) || this.chart1.startMove) {
                this.chart2.startMove = this.chart1.startMove;
                this.chart2.showCrosshair(args.x, args.y);
            }
        };
        public chart1MouseUp(args: IMouseEventArgs): void {

```

```

        if (Browser.isDevice && this.chart1.startMove) {
            this.chart2.hideCrosshair();
        }
    };
    public chart2MouseLeave(args: IMouseEventArgs): void {
        this.chart1.hideCrosshair();
    };
    public chart2MouseMove(args: IMouseEventArgs): void {
        if ((!Browser.isDevice && !this.chart2.isTouch &&
!this.chart2.isChartDrag) || this.chart2.startMove) {
            this.chart2.startMove = this.chart1.startMove;
            this.chart1.showCrosshair(args.x, args.y);
        }
    };
    public chart2MouseUp(args: IMouseEventArgs): void {
        if (Browser.isDevice && this.chart2.startMove) {
            this.chart1.hideCrosshair();
        }
    };
    ngOnInit(): void {
        this.chartData = synchronizedData;
        this.primaryXAxis = {
            minimum: new Date(2023, 1, 18),
            maximum: new Date(2023, 7, 18),
            valueType: 'DateTime',
            labelFormat: 'MMM d',
            lineStyle: { width: 0 },
            majorGridLines: { width: 0 },
            edgeLabelPlacement: Browser.isDevice ? 'None' : 'Shift',
            labelRotation: Browser.isDevice ? -45 : 0,
            interval: Browser.isDevice ? 2 : 1,
            crosshairTooltip: { enable: true },
        };
        this.primaryYAxis1 = {
            labelFormat: 'n2',
            majorTickLines: { width: 0 },
            lineStyle: { width: 0 },
            minimum: 0.86,
            maximum: 0.96,
            interval: 0.025
        };
        this.primaryYAxis2 = {
            labelFormat: 'n1',
            majorTickLines: { width: 0 },
            lineStyle: { width: 0 },
            minimum: 79,
            maximum: 85,
            interval: 1.5
        };
        this.title1 = 'US to Euro';
        this.title2 = 'US to INR';
        this.border = {
            width: 2
        };
        this.width = 2;
        this.titleStyle = {
            textAlignment: 'Near'
        };
    }

```

```

    };
    this.crosshair = {
        enable: true, lineType: 'Vertical', dashArray: '2,2'
    };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Zooming synchronization

You can maintain constant zoom levels across multiple charts using the [zoomComplete](#) event. In the [zoomComplete](#) event, obtain the [zoomFactor](#) and [zoomPosition](#) values of the particular chart, and then apply those values to the other charts.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, SplineAreaSeriesService } from '@syncfusion/ej2-angular-charts'
import { ZoomService } from '@syncfusion/ej2-angular-charts'
import { Component, ViewChild, OnInit } from '@angular/core';
import { IMouseEventArgs, ChartComponent, IZoomCompleteEventArgs } from '@syncfusion/ej2-angular-charts';
import { synchronizedData } from './datasource';
import { Browser } from '@syncfusion/ej2-base';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, SplineAreaSeriesService, ZoomService ],
  standalone: true,
  selector: 'app-container',
  template: `<div class="control-section">
    <div class="row">
      <div class="col" >
        <ejs-chart #chart1 style='display:block;' id="container1"
        [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis1'
        [title]='title1' [titleStyle]="titleStyle"
        [zoomSettings]='zoomSettings' (zoomComplete)="zoomComplete($event)">
          <e-series-collection>
            <e-series [dataSource]='chartData' type='Line'
            xName='USD' yName='EUR' [width]="width">
          </e-series>
        </e-series-collection>
      </ejs-chart>
    </div>
    <div class="col" >

```



```

        <ejs-chart #chart2 style='display:block;' id="container2"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis2'
        [title]='title2' [titleStyle]="titleStyle"
[zoomSettings]='zoomSettings' (zoomComplete)='zoomComplete($event)'>
        <e-series-collection>
            <e-series [dataSource]='chartData' type='SplineArea'
xName='USD' yName='INR' opacity=0.6
                [width]="width" [border]='border'>
            </e-series>
        </e-series-collection>
    </ejs-chart>
</div>
</div>
</div>`
}))
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public primaryYAxis1?: Object;
    public primaryYAxis2?: Object;
    public chartData?: Object[];
    public title1?: string;
    public title2?: string;
    public titleStyle?: Object;
    public border?: Object;
    public width?: number;
    public zoomSettings?: Object;
    @ViewChild('chart1')
    public chart1!: ChartComponent;
    @ViewChild('chart2')
    public chart2!: ChartComponent;
    public charts: ChartComponent[] = [];
    public zoomFactor: number = 0;
    public zoomPosition: number = 0;
    ngAfterViewInit() {
        this.charts = [this.chart1, this.chart2];
    }
    public chart1MouseLeave(args: IMouseEventArgs): void {
        this.chart2.hideTooltip();
    };
    public zoomComplete(args: IZoomCompleteEventArgs): void {
        if (args.axis.name === 'primaryXAxis') {
            this.zoomFactor = args.currentZoomFactor;
            this.zoomPosition = args.currentZoomPosition;
            this.zoomCompleteFunction(args);
        }
    };
    public zoomCompleteFunction(args: any): void {
        for (let i: number = 0; i < this.charts.length; i++) {
            if (args.axis.series[0].chart.element.id !==
this.charts[i].element.id) {
                this.charts[i].primaryXAxis.zoomFactor = this.zoomFactor;
                this.charts[i].primaryXAxis.zoomPosition =
this.zoomPosition;
                this.charts[i].zoomModule.isZoomed =
args.axis.series[0].chart.zoomModule.isZoomed;
                this.charts[i].zoomModule.isPanning =
args.axis.series[0].chart.zoomModule.isPanning;
            }
        }
    }
}

```

```

    }
  }
};
ngOnInit(): void {
  this.chartData = synchronizedData;
  this.primaryXAxis = {
    minimum: new Date(2023, 1, 18),
    maximum: new Date(2023, 7, 18),
    valueType: 'DateTime',
    labelFormat: 'MMM d',
    lineStyle: { width: 0 },
    majorGridLines: { width: 0 },
    edgeLabelPlacement: Browser.isDevice ? 'None' : 'Shift',
    labelRotation: Browser.isDevice ? -45 : 0,
    interval: Browser.isDevice ? 2 : 1
  };
  this.primaryYAxis1 = {
    labelFormat: 'n2',
    majorTickLines: { width: 0 },
    lineStyle: { width: 0 },
    minimum: 0.86,
    maximum: 0.96,
    interval: 0.025
  };
  this.primaryYAxis2 = {
    labelFormat: 'n1',
    majorTickLines: { width: 0 },
    lineStyle: { width: 0 },
    minimum: 79,
    maximum: 85,
    interval: 1.5
  };
  this.title1 = 'US to Euro';
  this.title2 = 'US to INR';
  this.zoomSettings = {
    enableMouseWheelZooming: true,
    enablePinchZooming: true,
    enableScrollbar: false,
    enableDeferredZooming: false,
    enableSelectionZooming: true,
    enablePan: true,
    mode: 'X',
    toolbarItems: ['Pan', 'Reset']
  };
  this.border = {
    width: 2
  };
  this.width = 2;
  this.titleStyle = {
    textAlignment: 'Near'
  };
}
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Selection synchronization

You can select the data across multiple charts using the [selectionComplete](#) event. In the `selectionComplete` event, obtain the selected values of the particular chart, and then apply those values to the other charts.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, SplineSeriesService } from '@syncfusion/ej2-angular-charts'
import { SelectionService, ZoomService } from '@syncfusion/ej2-angular-charts'
import { Component, ViewChild, OnInit } from '@angular/core';
import { IMouseEventArgs, ChartComponent, IZoomCompleteEventArgs, ISelectionCompleteEventArgs } from '@syncfusion/ej2-angular-charts';
import { synchronizedData } from './datasource';
import { Browser } from '@syncfusion/ej2-base';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ DateTimeService, LineSeriesService, SplineSeriesService, SelectionService, ZoomService ],
  standalone: true,
  selector: 'app-container',
  template: `<div class="control-section">
<div class="row">
  <div class="col" >
    <ejs-chart #chart1 style='display:block;' id="container1"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis1'
[title]='title1' [titleStyle]="titleStyle"
[zoomSettings]='zoomSettings' (zoomComplete)="zoomComplete($event)"
selectionMode='Point' selectionPattern='Box'
(selectionComplete)="selectionComplete($event)">
      <e-series-collection>
        <e-series [dataSource]='chartData' type='Line'
xName='USD' yName='EUR' [width]="width">
      </e-series>
    </e-series-collection>
  </ejs-chart>
</div>
<div class="col" >
    <ejs-chart #chart2 style='display:block;' id="container2"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis2'
[title]='title2' [titleStyle]="titleStyle"
[zoomSettings]='zoomSettings' (zoomComplete)="zoomComplete($event)"
selectionMode='Point' selectionPattern='Box'
(selectionComplete)="selectionComplete($event)">
      <e-series-collection>
```

```

        <e-series [dataSource]='chartData' type='Spline'
        xName='USD' yName='INR'
            [width]="width" [border]='border'>
        </e-series>
    </e-series-collection>
</ejs-chart>
</div>
</div>
</div>`
}))
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public primaryYAxis1?: Object;
    public primaryYAxis2?: Object;
    public chartData?: Object[];
    public title1?: string;
    public title2?: string;
    public titleStyle?: Object;
    public border?: Object;
    public width?: number;
    public zoomSettings?: Object;
    @ViewChild('chart1')
    public chart1!: ChartComponent;
    @ViewChild('chart2')
    public chart2!: ChartComponent;
    public charts: ChartComponent[] = [];
    public zoomFactor: number = 0;
    public zoomPosition: number = 0;
    public count: number = 0;
    ngAfterViewInit() {
        this.charts = [this.chart1, this.chart2];
    }
    public zoomComplete(args: IZoomCompleteEventArgs): void {
        if (args.axis.name === 'primaryXAxis') {
            this.zoomFactor = args.currentZoomFactor;
            this.zoomPosition = args.currentZoomPosition;
            this.zoomCompleteFunction(args);
        }
    };
    public zoomCompleteFunction(args: any): void {
        for (let i: number = 0; i < this.charts.length; i++) {
            if (args.axis.series[0].chart.element.id !==
this.charts[i].element.id) {
                this.charts[i].primaryXAxis.zoomFactor = this.zoomFactor;
                this.charts[i].primaryXAxis.zoomPosition =
this.zoomPosition;
                this.charts[i].zoomModule.isZoomed =
args.axis.series[0].chart.zoomModule.isZoomed;
                this.charts[i].zoomModule.isPanning =
args.axis.series[0].chart.zoomModule.isPanning;
            }
        }
    };
    public selectionComplete(args: ISelectionCompleteEventArgs): void {
        this.selectionCompleteFunction(args);
    }
    public selectionCompleteFunction(args: any): void {

```

```

        if (this.count == 0) {
            for (var j = 0; j < args.selectedDataValues.length; j++) {
                args.selectedDataValues[j].point =
args.selectedDataValues[j].pointIndex;
                args.selectedDataValues[j].series =
args.selectedDataValues[j].seriesIndex;
            }
            for (var i = 0; i < this.charts.length; i++) {
                if (args.chart.element.id !== this.charts[i].element.id) {
                    this.charts[i].selectedDataIndexes =
args.selectedDataValues;
                    this.count += 1;
                    this.charts[i].dataBind();
                }
            }
            this.count = 0;
        }
    }
    ngOnInit(): void {
        this.chartData = synchronizedData;
        this.primaryXAxis = {
            minimum: new Date(2023, 1, 18),
            maximum: new Date(2023, 7, 18),
            valueType: 'DateTime',
            labelFormat: 'MMM d',
            lineStyle: { width: 0 },
            majorGridLines: { width: 0 },
            edgeLabelPlacement: Browser.isDevice ? 'None' : 'Shift',
            labelRotation: Browser.isDevice ? -45 : 0,
            interval: Browser.isDevice ? 2 : 1
        };
        this.primaryYAxis1 = {
            labelFormat: 'n2',
            majorTickLines: { width: 0 },
            lineStyle: { width: 0 },
            minimum: 0.86,
            maximum: 0.96,
            interval: 0.025
        };
        this.primaryYAxis2 = {
            labelFormat: 'n1',
            majorTickLines: { width: 0 },
            lineStyle: { width: 0 },
            minimum: 79,
            maximum: 85,
            interval: 1.5
        };
        this.title1 = 'US to Euro';
        this.title2 = 'US to INR';
        this.zoomSettings = {
            enableSelectionZooming: true,
            mode: 'X'
        };
        this.border = {
            width: 2
        };
        this.width = 2;
    }

```

```

        this.titleStyle = {
            textAlign: 'Near'
        };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

Selection in Angular Chart component

Chart provides selection support for the series and its data points on mouse click.

When Mouse is clicked on the data points, the corresponding series legend will also be selected.

We have different type of selection mode for selecting the data. They are,

- None
- Point
- Series
- Cluster
- DragXY
- DragX
- DragY

Point

You can select a point, by setting `selectionMode` to point.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChartModule } from '@syncfusion/ej2-angular-charts';
import { CategoryService, ColumnSeriesService } from '@syncfusion/ej2-angular-charts';
import { LegendService, SelectionService } from '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
import { selectionData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, ColumnSeriesService, LegendService, SelectionService ],
  standalone: true,
  selector: 'app-container',
})

```

```

    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
selectionMode='Point'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver'></e-series>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' ></e-series>
    </e-series-collection>
</ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    ngOnInit(): void {
      this.chartData = selectionData;
      this.primaryXAxis = {
        valueType: 'Category',
        title: 'Countries'
      };
      this.title = 'Olympic Medals';
    }
  }
}

```

APP.COMPONENT.CSS

```

#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 16px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.animation {
  background: #333333;
  border: 1px solid #cecece;
  box-sizing: border-box;
  height: 100px;
  width: 100px;
}
#chart-container {
  display: block;
  height: 350px;
}
.chartSelection1 {
  fill: red;
}
.chartSelection2 {
  fill: green;
}

```

```
.chartSelection3 {
    fill: blue;
}
```

Note: To use select feature, we need to Inject `SelectionService` into the `NgModule.providers`.

Series

You can select a series, by setting `selectionMode` to series.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { selectionData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, ColumnSeriesService, LegendService,
    SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
    selectionMode='Series'>
      <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='country'
        yName='gold' name='Gold' ></e-series>
        <e-series [dataSource]='chartData' type='Column' xName='country'
        yName='silver' name='Silver'></e-series>
        <e-series [dataSource]='chartData' type='Column' xName='country'
        yName='bronze' name='Bronze' ></e-series>
      </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = selectionData;
    this.primaryXAxis = {
      valueType: 'Category'
    };
    this.title = 'Olympic Medals';
  }
}
```


APP.COMPONENT.CSS

```
#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 16px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.animation {
  background: #333333;
  border: 1px solid #cecece;
  box-sizing: border-box;
  height: 100px;
  width: 100px;
}
#chart-container {
  display: block;
  height: 350px;
}
.chartSelection1 {
  fill: red;
}
.chartSelection2 {
  fill: green;
}
.chartSelection3 {
  fill: blue;
}
```

Cluster

You can select the points that corresponds to the same index in all the series, by setting `selectionMode` to cluster.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { selectionData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, ColumnSeriesService, LegendService,
    SelectionService ],
  standalone: true,
  selector: 'app-container',
```

```

    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
selectionMode='Cluster'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver'></e-series>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' ></e-series>
    </e-series-collection>
</ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    ngOnInit(): void {
      this.chartData = selectionData;
      this.primaryXAxis = {
        valueType: 'Category',
        title: 'Countries'
      };
      this.title = 'Olympic Medals';
    }
  }

```

APP.COMPONENT.CSS

```

#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 16px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.animation {
  background: #333333;
  border: 1px solid #cecece;
  box-sizing: border-box;
  height: 100px;
  width: 100px;
}
#chart-container {
  display: block;
  height: 350px;
}
.chartSelection1 {
  fill: red;
}
.chartSelection2 {
  fill: green;
}

```

```

}
.chartSelection3 {
  fill: blue;
}

```

Rectangular selection

DragXY, DragX and DragY

To fetch the collection of data under a particular region, you have to set `selectionMode` as `DragXY`.

- DragXY - Allows us to select data with respect to horizontal and vertical axis.
- DragX - Allows us to select data with respect to horizontal axis.
- DragY - Allows us to select data with respect to vertical axis.

The selected data's are returned as an array collection in the `[dragComplete]`

(<https://ej2.syncfusion.com/angular/documentation/api/chart/iDragCompleteEventArgs/>) event.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ScatterSeriesService, LegendService, SelectionService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { ChartData } from './chartdata.service';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ ScatterSeriesService, LegendService, SelectionService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
selectionMode='DragXY'>
    <e-series-collection>
      <e-series [dataSource]='series1' type='Scatter' xName='x'
yName='y' name='Male' opacity=0.7 [marker]='marker'></e-series>
      <e-series [dataSource]='series2' type='Scatter' xName='x'
yName='y' name='Female' opacity=0.7 [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public series1?: Object;
  public series2?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.primaryXAxis = {

```

```

        title: 'Height (cm)',
        minimum: 120, maximum: 180,
        edgeLabelPlacement: 'Shift',
        labelFormat: '{value}cm'
    };
    this.primaryYAxis = {
        title: 'Weight (kg)',
        minimum: 60, maximum: 90,
        labelFormat: '{value}kg',
        rangePadding: 'None'
    };
    this.title = 'Height Vs Weight';
    this.marker = { width: 10, height: 10 };
    this.series1 = ChartData.prototype.getScatterData().series1;
    this.series2 = ChartData.prototype.getScatterData().series2;
    }
}

```

APP.COMPONENT.CSS

```

#loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 16px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.animation {
    background: #333333;
    border: 1px solid #cecece;
    box-sizing: border-box;
    height: 100px;
    width: 100px;
}
#chart-container {
    display: block;
    height: 350px;
}

```

Lasso selection

To select a region by drawing freehand shapes to fetch a collection of data use `selectionMode` as `Lasso`. You can also select multiple regions on the chart through this mode.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ScatterSeriesService, LegendService, SelectionService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { ChartData } from '../chartdata.service';

```

```

@Component({
  imports: [
    ChartModule
  ],
  providers: [ ScatterSeriesService, LegendService, SelectionService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
selectionMode='Lasso'>
    <e-series-collection>
      <e-series [dataSource]='series1' type='Scatter' xName='x'
yName='y' name='Male' opacity=0.7 [marker]='marker'></e-series>
      <e-series [dataSource]='series2' type='Scatter' xName='x'
yName='y' name='Female' opacity=0.7 [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public series1?: Object;
  public series2?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.primaryXAxis = {
      title: 'Height (cm)',
      minimum: 120, maximum: 180,
      edgeLabelPlacement: 'Shift',
      labelFormat: '{value}cm'
    };
    this.primaryYAxis = {
      title: 'Weight (kg)',
      minimum: 60, maximum: 90,
      labelFormat: '{value}kg',
      rangePadding: 'None'
    };
    this.title = 'Height Vs Weight';
    this.marker = { width: 10, height: 10 };
    this.series1 = ChartData.prototype.getScatterData().series1;
    this.series2 = ChartData.prototype.getScatterData().series2;
  }
}

```

APP.COMPONENT.CSS

```

#loader {
  color: #008cff;
  font-family: 'Helvetica Neue', 'calibiri';
  font-size: 16px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
}

```

```

        width: 30%;
    }
    .animation {
        background: #333333;
        border: 1px solid #cecece;
        box-sizing: border-box;
        height: 100px;
        width: 100px;
    }
    #chart-container {
        display: block;
        height: 350px;
    }
}

```

Multi-region selection

To select multiple region on the chart, set the `allowMultiSelection` property to true.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ScatterSeriesService, LegendService, SelectionService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { ChartData } from './chartdata.service';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ ScatterSeriesService, LegendService, SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
selectionMode='DragXY' allowMultiSelection='true'>
    <e-series-collection>
      <e-series [dataSource]='series1' type='Scatter' xName='x'
yName='y' name='Male' opacity=0.7 [marker]='marker'></e-series>
      <e-series [dataSource]='series2' type='Scatter' xName='x'
yName='y' name='Female' opacity=0.7 [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public series1?: Object;
  public series2?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.primaryXAxis = {
      title: 'Height (cm)',
      minimum: 120, maximum: 180,

```

```

        edgeLabelPlacement: 'Shift',
        labelFormat: '{value}cm'
    };
    this.primaryYAxis = {
        title: 'Weight (kg)',
        minimum: 60, maximum: 90,
        labelFormat: '{value}kg',
        rangePadding: 'None'
    };
    this.title = 'Height Vs Weight';
    this.marker = { width: 10, height: 10 };
    this.series1 = ChartData.prototype.getScatterData().series1;
    this.series2 = ChartData.prototype.getScatterData().series2;
}
}

```

APP.COMPONENT.CSS

```

#loader {
    color: #008c8f;
    font-family: 'Helvetica Neue','calibiri';
    font-size:16px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.animation {
    background: #333333;
    border: 1px solid #cecece;
    box-sizing: border-box;
    height: 100px;
    width: 100px;
}
#chart-container {
    display: block;
    height: 350px;
}

```

Selection type

You can select multiple points or series, by enabling the [isMultiSelect](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService } from '@syncfusion/ej2-
angular-charts'
import { LegendService, SelectionService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { selectionData } from './datasource';
@Component({
    imports: [

```

```

    ChartModule
  ],
  providers: [ CategoryService, ColumnSeriesService, LegendService,
    SelectionService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
    selectionMode='Point' isMultiSelect='true'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' ></e-series>
    </e-series-collection>
  </ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    ngOnInit(): void {
      this.chartData = selectionData;
      this.primaryXAxis = {
        valueType: 'Category',
        title: 'Countries'
      };
      this.title = 'Olympic Medals';
    }
  }
}

```

APP.COMPONENT.CSS

```

#loader {
  color: #008c8f;
  font-family: 'Helvetica Neue','calibiri';
  font-size:16px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.animation {
  background: #333333;
  border: 1px solid #cecece;
  box-sizing: border-box;
  height: 100px;
  width: 100px;
}
#chart-container {
  display: block;
  height: 350px;
}

```



```

}
.chartSelection1 {
    fill:red;
}
.chartSelection2 {
    fill: green;
}
.chartSelection3 {
    fill: blue;
}

```

Selection on load

You can able to select a point or series programmatically on a chart using [selectedDataIndexes](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService } from '@syncfusion/ej2-
angular-charts'
import { LegendService, SelectionService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { selectionData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, ColumnSeriesService, LegendService,
  SelectionService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
selectionMode='Point' isMultiSelect='true'
[selectedDataIndexes]='selectedData'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' selectionStyle='chartSelection1'
[animation]='animation'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver' selectionStyle='chartSelection2'
[animation]='animation'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' selectionStyle='chartSelection3'
[animation]='animation'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public animation?: Object;

```

```

public selectedData?: Object[];
ngOnInit(): void {
  this.chartData = selectionData;
  this.primaryXAxis = {
    valueType: 'Category',
    title: 'Countries'
  };
  this.animation = { enable: false };
  this.selectedData = [
    { series: 0, point: 1 }, { series: 2, point: 3 }
  ];
  this.title = 'Olympic Medals';
}
}

```

APP.COMPONENT.CSS

```

#loader {
  color: #008cff;
  font-family: 'Helvetica Neue','calibiri';
  font-size:16px;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
.animation {
  background: #333333;
  border: 1px solid #cecece;
  box-sizing: border-box;
  height: 100px;
  width: 100px;
}
#chart-container {
  display: block;
  height: 350px;
}
.chartSelection1 {
  fill:red;
}
.chartSelection2 {
  fill: green;
}
.chartSelection3 {
  fill: blue;
}

```

Selection through on legend

You can able to select a point or series through on legend using [toggleVisibility](#) property. Also, use [enableHighlight](#) property for highlighting the series through legend.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'

```

```

import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { selectionData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, ColumnSeriesService, LegendService, SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[legendSettings]='legendSettings' selectionMode='Point'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' selectionStyle='chartSelection1'
[animation]='animation'></e-series>
    <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver' selectionStyle='chartSelection2'
[animation]='animation'></e-series>
    <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' selectionStyle='chartSelection3'
[animation]='animation'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public animation?: Object;
  public selectedData?: Object[];
  public legendSettings?: Object;
  ngOnInit(): void {
    this.chartData = selectionData;
    this.primaryXAxis = {
      valueType: 'Category',
      title: 'Countries'
    };
    this.animation = { enable: false };
    this.legendSettings = { visible: true, toggleVisibility: false,
enableHighlight: true };
    this.title = 'Olympic Medals';
  }
}

```

APP.COMPONENT.CSS

```
#loader {
```

```

        color: #008cff;
        font-family: 'Helvetica Neue','calibiri';
        font-size:16px;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    .animation {
        background: #333333;
        border: 1px solid #cecece;
        box-sizing: border-box;
        height: 100px;
        width: 100px;
    }
    #chart-container {
        display: block;
        height: 350px;
    }
    .chartSelection1 {
        fill:red;
    }
    .chartSelection2 {
        fill: green;
    }
    .chartSelection3 {
        fill: blue;
    }
}

```

Customization for selection

You can apply custom style to selected points or series with [selectionStyle](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { selectionData } from './datasource';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, ColumnSeriesService, LegendService,
    SelectionService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
selectionMode='Point' isMultiSelect='true'>
    <e-series-collection>

```

```

        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' selectionStyle='chartSelection1'></e-series>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver' selectionStyle='chartSelection2'></e-series>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' selectionStyle='chartSelection3'></e-series>
    </e-series-collection>
</ejs-chart>`
    ))
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    ngOnInit(): void {
        this.chartData = selectionData;
        this.primaryXAxis = {
            valueType: 'Category',
            title: 'Countries'
        };
        this.title = 'Olympic Medals';
    }
}

```

APP.COMPONENT.CSS

```

#loader {
    color: #008cff;
    font-family: 'Helvetica Neue','calibiri';
    font-size:16px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
.animation {
    background: #333333;
    border: 1px solid #cecece;
    box-sizing: border-box;
    height: 100px;
    width: 100px;
}
#chart-container {
    display: block;
    height: 350px;
}
.chartSelection1 {
    fill:red;
}
.chartSelection2 {
    fill: green;
}
.chartSelection3 {
    fill: blue;
}

```

See Also

- [Display selected data for range selection](#)

Chart print in Angular Chart component

Print

The rendered chart can be printed directly from the browser by calling the public method print.

You can pass array of ID of elements or element to this method. By default it take element of the chart.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, ILoadedEventArgs, SplineSeriesService}
from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation, ViewChild } from
'@angular/core';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
import { ChartComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    ChartModule, ButtonModule, ChartAllModule
  ],
  providers: [ AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<div class="col-md-8">
<button ej-button id='print' (click)= 'print()'>Print</button>
<ejs-chart #chart id='chart-container' [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis'
[title]='title' >
<e-series-collection>
<e-series [dataSource]='data' type='Radar' xName='x'
yName='y' drawType='Line'> </e-series>
</e-series-collection>
</ejs-chart>
</div> `
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public primaryYAxis?: Object;
  public data?: Object[];
  @ViewChild('chart')
  public chartObj?: ChartComponent;
```

```

ngOnInit(): void {
    this.data = [
        { x: 2005, y: 28 }, { x: 2006, y: 25 }, { x: 2007, y: 26 }, { x:
2008, y: 27 },
        { x: 2009, y: 32 }, { x: 2010, y: 35 }, { x: 2011, y: 30 }
    ];
    this.primaryXAxis = {
        title: 'Year', coefficient: 90,
        minimum: 2004, maximum: 2012, interval: 1
    };
    this.primaryYAxis = {
        minimum: 20, maximum: 40, interval: 5,
        title: 'Efficiency',
        labelFormat: '{value}%'
    };
    this.title = 'Efficiency of oil-fired power production';
}
print() {
    this.chartObj?.print();
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Export

The rendered chart can be exported to JPEG, PNG, SVG, PDF, XLSX, or CSV format using the export method in chart. The input parameters for this method are type for format and fileName for result.

The optional parameters for this method are,

- orientation - either portrait or landscape mode during PDF export,
- controls - pass collections of controls for multiple export,
- width - width of chart export, and
- height - height of chart export.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, ILoadedEventArgs, SplineSeriesService}
from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { ILoadedEventArgs } from '@syncfusion/ej2-angular-charts';

```

```

@Component({
  imports: [
    ChartModule, ButtonModule, ChartAllModule
  ],
  providers: [ AreaSeriesService, LineSeriesService, ExportService,
    ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
    RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
    CategoryService, RadarSeriesService, SplineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<div class="col-md-8">
    <ejs-chart id='chart-container' [primaryXAxis]='primaryXAxis'
    [primaryYAxis]='primaryYAxis' (loaded)='loaded($event)'
      [title]='title' >
        <e-series-collection>
          <e-series [dataSource]='data' type='Radar' xName='x'
yName='y' drawType='Line'> </e-series>
        </e-series-collection>
      </ejs-chart>
    </div> `
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public primaryYAxis?: Object;
  public data?: Object[];
  public loaded: Function | any;
  ngOnInit(): void {
    this.data = [{ x: 2005, y: 28 }, { x: 2006, y: 25 }, { x: 2007, y: 26
}, { x: 2008, y: 27 },
    { x: 2009, y: 32 }, { x: 2010, y: 35 }, { x: 2011, y:
30 }];
    this.primaryXAxis = {
      title: 'Year', coefficient: 90,
      minimum: 2004, maximum: 2012, interval: 1
    };
    this.primaryYAxis = {
      minimum: 20, maximum: 40, interval: 5,
      title: 'Efficiency',
      labelFormat: '{value}%'
    };
    this.title = 'Efficiency of oil-fired power production';
    this.loaded = (args: ILoadedEventArgs) => {
      args.chart.exportModule.export('PNG', 'export');
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```


Adding header and footer in PDF export

In the export method, specify the following parameters to add a header and footer text to the exported PDF document:

- **header** - Specify the text that should appear at the top of the exported PDF document.
- **footer** - Specify the text that should appear at the bottom of the exported PDF document.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, ILoadedEventArgs, SplineSeriesService}
from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { ChartComponent } from '@syncfusion/ej2-angular-charts';
@Component({
imports: [
    ChartModule, ButtonModule, ChartAllModule
],
providers: [ AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService],
standalone: true,
selector: 'app-container',
template: `<ejs-chart #chart id='chart-container'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[title]='title' >
    <e-series-collection>
        <e-series [dataSource]='exportData' type='Column' xName='x'
yName='y' width=2> </e-series>
    </e-series-collection>
</ejs-chart>
<button ej-button id='print' (click)='export()'>Export</button>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public title?: string;
    public primaryYAxis?: Object;
    public exportData?: Object[];
    @ViewChild('chart')
    public chart?: ChartComponent;
    ngOnInit(): void {
        this.exportData = [{ x: 'John', y: 10000 }, { x: 'Jake', y: 12000 },
{ x: 'Peter', y: 18000 },
{ x: 'James', y: 11000 }, { x: 'Mary', y: 9700 }];
        this.primaryXAxis = {
            title: 'Manager',
            valueType: 'Category',
            majorGridLines: { width: 0 }
        }
    }
}
```

```

    };
    this.primaryYAxis = {
      title: 'Sales',
      minimum: 0,
      maximum: 20000,
      majorGridLines: { width: 0 }
    };
    this.title = 'Sales Comparision';
  }
  export() {
    const header = {
      content: 'Chart Header',
      fontSize: 15
    };
    const footer = {
      content: 'Chart Footer',
      fontSize: 15,
    };
    this.chart?.exportModule.export('PDF', 'Chart', 1, [this.chart as
ChartComponent], undefined, undefined, true, header, footer);
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Exporting charts into separate page during the PDF export

During PDF export, set the `exportToMultiplePage` parameter to `true` to export each chart as a separate page.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartModule } from
'@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, ILoadedEventArgs, SplineSeriesService,
AccumulationLegendService, AccumulationTooltipService,
AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { ChartComponent } from '@syncfusion/ej2-angular-charts';
import { Chart } from '@syncfusion/ej2-charts';
@Component({
  imports: [
    ChartModule, ButtonModule, ChartAllModule, AccumulationChartModule
  ],

```

```

providers: [ AreaSeriesService, LineSeriesService, ExportService,
ColumnSeriesService, StackingColumnSeriesService, StackingAreaSeriesService,
RangeColumnSeriesService, ScatterSeriesService, PolarSeriesService,
CategoryService, RadarSeriesService, SplineSeriesService,
AccumulationLegendService, AccumulationTooltipService,
AccumulationDataLabelService],
standalone: true,
  selector: 'app-container',
  template: `<ejs-chart #chart id='chart-container1'
[primaryXAxis]='primaryXAxis1' [primaryYAxis]='primaryYAxis1'
  [title]='title1' >
    <e-series-collection>
      <e-series [dataSource]='data1' type='Line' xName='x'
yName='y' width=2 name='Germany' [marker]='marker'> </e-series>
      <e-series [dataSource]='data1' type='Line' xName='x'
yName='y1' width=2 name='England' [marker]='marker'> </e-series>
    </e-series-collection>
  </ejs-chart>
  <ejs-chart #chart1 id='chart-container2' [primaryXAxis]='primaryXAxis2'
[primaryYAxis]='primaryYAxis2'
  [title]='title2' >
    <e-series-collection>
      <e-series [dataSource]='data2' type='Column' xName='x'
yName='y' width=2> </e-series>
    </e-series-collection>
  </ejs-chart>
  <ejs-accumulationchart #chart2 id="chart-container3"
[legendSettings]='legendSettings' [tooltip]='tooltip'
[enableSmartLabels]='true'>
    <e-accumulation-series-collection>
      <e-accumulation-series [dataSource]='data3' xName='x' yName='y'
[dataLabel]='datalabel' radius='70%' startAngle=0 endAngle=360
name='Project'></e-accumulation-series>
    </e-accumulation-series-collection>
  </ejs-accumulationchart>

  <button ej-button id='print' (click)='export()'>Export</button>`
})
export class AppComponent implements OnInit {
  public primaryXAxis1?: Object;
  public primaryXAxis2?: Object;
  public primaryXAxis3?: Object;
  public primaryYAxis1?: Object;
  public primaryYAxis2?: Object;
  public primaryYAxis3?: Object;
  public title1?: string;
  public title2?: string;
  public title3?: string;
  public data1?: Object[];
  public data2?: Object[];
  public data3?: Object[];
  public marker?: Object;
  public legendSettings?: Object;
  public tooltip?: Object;
  public datalabel?: Object;
  @ViewChild('chart')
  public chart?: ChartComponent;

```

```

    @ViewChild('chart1')
    public chart1?: ChartComponent;
    @ViewChild('chart2')
    public chart2?: ChartComponent;
    ngOnInit(): void {
        this.data1 = [
            { x: new Date(2005, 0, 1), y: 21, y1: 28 }, { x: new Date(2006,
0, 1), y: 24, y1: 44 },
            { x: new Date(2007, 0, 1), y: 36, y1: 48 }, { x: new Date(2008,
0, 1), y: 38, y1: 50 },
            { x: new Date(2009, 0, 1), y: 54, y1: 66 }, { x: new Date(2010,
0, 1), y: 57, y1: 78 },
            { x: new Date(2011, 0, 1), y: 70, y1: 84 }
        ];
        this.data2 = [
            { x: 'John', y: 10000 }, { x: 'Jake', y: 12000 }, { x: 'Peter',
y: 18000 },
            { x: 'James', y: 11000 }, { x: 'Mary', y: 9700 }
        ];
        this.data3 = [
            { x: 'Labour', y: 18, text: '18%' }, { x: 'Legal', y: 8, text:
'8%' },
            { x: 'Production', y: 15, text: '15%' }, { x: 'License', y: 11,
text: '11%' },
            { x: 'Facilities', y: 18, text: '18%' }, { x: 'Taxes', y: 14,
text: '14%' },
            { x: 'Insurance', y: 16, text: '16%' }
        ];
        this.primaryXAxis1 = {
            valueType: 'DateTime',
            labelFormat: 'y',
            intervalType: 'Years',
            edgeLabelPlacement: 'Shift',
            majorGridLines: { width: 0 }
        };
        this.primaryYAxis1 = {
            labelFormat: '{value}%',
            rangePadding: 'None',
            minimum: 0,
            maximum: 100,
            interval: 20,
            lineStyle: { width: 0 },
            majorTickLines: { width: 0 },
            minorTickLines: { width: 0 }
        };
        this.title1 = 'Medal Count';
        this.marker = { visible: true, width: 10, height: 10 };
        this.primaryXAxis2 = {
            title: 'Manager',
            valueType: 'Category',
            majorGridLines: { width: 0 }
        };
        this.primaryYAxis2 = {
            title: 'Sales',
            minimum: 0,
            maximum: 20000,
            majorGridLines: { width: 0 }
        };
    }

```

```

    };
    this.title2 = 'Sales Comparision';
    this.title3 = 'Project Cost Breakdown';
    this.legendSettings = {
        visible: true
    };
    this.tooltip = {
        enable: false
    };
    this.datalabel = { visible: true, name: 'text', position: 'Inside',
font: { fontWeight: '600', color: '#ffffff' } };
    }
    export() {
        this.chart?.exportModule.export('PDF', 'Chart', undefined,
[ this.chart as Chart, this.chart1 as Chart, this.chart2 as Chart],
undefined, undefined, true, undefined, undefined, true);
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Multiple Chart Export

You can export the multiple charts in single page by passing the multiple chart objects in the export method of chart. To export multiple charts in a single page, follow the given steps:

Initially, render more than one chart to export, and then add button to export the multiple charts. In button click, call the export method in charts, and then pass the multiple chart objects in the export method.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CategoryService, ColumnSeriesService, ExportService, LegendService,
DataLabelService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { ChartComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    ChartModule, ButtonModule
  ],
  providers: [ CategoryService, ColumnSeriesService, ExportService,
LegendService, DataLabelService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart #chart id='chartcontainer'
    [title]='title' (loaded)='loaded($event)'>
    <e-series-collection>

```

```

        <e-series [dataSource]='data' type='Column' xName='x'
yName='y' > </e-series>
    </e-series-collection>
</ejs-chart>
<ejs-chart #chart1 id='chartcontainer1'
    [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='data1' type='Column' xName='x'
yName='y' > </e-series>
    </e-series-collection>
</ejs-chart>
<button ej-button id='print' (click)='export()'>Export</button>`
    })
    export class AppComponent implements OnInit {
        public title?: string;
        public data?: Object[];
        public data1?: Object[];
        @ViewChild('chart')
        public chart?: ChartComponent;
        @ViewChild('chart1')
        public chart1?: ChartComponent;
        ngOnInit(): void {
            this.data = [
                { x: 1, y: 20 }, { x: 2, y: 5 },
                { x: 3, y: 10 }, { x: 4, y: 40 }
            ];
            this.data1 = [
                { x: 1, y: 20 }, { x: 2, y: 5 },
                { x: 3, y: 10 }, { x: 4, y: 40 }
            ];
            this.title = 'Chart 1';
        }
        export() {
            this.chart?.exportModule.export('PNG', 'chart', 'Landscape' as any
, [this.chart, this.chart1 as ChartComponent]);
        }
        loaded(args: any) {
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Exporting chart using base64 string

The chart can be exported as an image in the form of a base64 string by utilizing HTML canvas. This process involves rendering the chart onto a canvas element and then converting the canvas content to a base64 string.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'

```

```

import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CategoryService, ColumnSeriesService, ExportService, LegendService,
DataLabelService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { ChartComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    ChartModule, ButtonModule
  ],
  providers: [ CategoryService, ColumnSeriesService, ExportService,
LegendService, DataLabelService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart #chart style='display:block;' id='chartcontainer'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[chartArea]='chartArea'>
  <e-series-collection>
    <e-series [dataSource]='data' type='Column' xName='x' yName='y'
width=2>
    </e-series>
  </e-series-collection>
</ejs-chart>
<button ej2-button iconCss="e-icons e-export-icon" cssClass="e-flat"
isPrimary=true (click)="onClick($event)" style="text-transform:none
!important" id="togglebtn">EXPORT</button>`
})
export class AppComponent implements OnInit {
  public data: Object[] = [
    { x: 'DEU', y: 35.5 }, { x: 'CHN', y: 18.3 }, { x: 'ITA', y: 17.6 },
    { x: 'JPN', y: 13.6 },
    { x: 'US', y: 12 }, { x: 'ESP', y: 5.6 }, { x: 'FRA', y: 4.6 }, { x:
'AUS', y: 3.3 },
    { x: 'BEL', y: 3 }, { x: 'UK', y: 2.9 }
  ];
  //Initializing Primary X Axis
  public primaryXAxis: Object = {
    valueType: 'Category',
    majorGridLines: { width: 0 },
    majorTickLines: { width: 0 },
    minorTickLines: { width: 0 }
  };
  //Initializing Primary Y Axis
  public primaryYAxis: Object = {
    title: 'Measurements',
    labelFormat: '{value}GW',
    minimum: 0,
    maximum: 40,
    interval: 10,
    lineStyle: {width : 0},
    minorTickLines: {width: 0},
    majorTickLines: {width : 0},
  };
  public chartArea: Object = {
    border: {
      width: 0
    }
  }
}

```

```

    }
};

public title: string = 'Top 10 Countries Using Solar Power';

public onClick(e: Event): void {
    let svg: any = document.querySelector("#chartcontainer_svg");
    var svgData = new XMLSerializer().serializeToString(svg);
    var canvas = document.createElement("canvas");
    document.body.appendChild(canvas);
    var svgSize = svg.getBoundingClientRect();
    canvas.width = svgSize.width;
    canvas.height = svgSize.height;
    let ctx: any = canvas.getContext("2d");
    var img = document.createElement("img");
    img.setAttribute("src", "data:image/svg+xml;base64," +
    btoa(svgData));
    img.onload = function() {
        ctx.drawImage(img, 0, 0);
        var imagedata = canvas.toDataURL("image/png");
        console.log(imagedata); // printed base64 in console
        canvas.remove();
    };
}

ngOnInit(): void {
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Chart appearance in Angular Chart component

Custom color palette

You can customize the default color of series or points by providing a custom color palette of your choice by using the [palettes](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],

```



```

providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService],
standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[title]='title' [palettes]='palette'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' ></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='silver' name='Silver'></e-series>
      <e-series [dataSource]='chartData' type='Column' xName='country'
yName='bronze' name='Bronze' ></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public palette?: string[];
  ngOnInit(): void {
    this.chartData = [
      { country: "USA", gold: 50, silver: 70, bronze: 45 },
      { country: "China", gold: 40, silver: 60, bronze: 55 },
      { country: "Japan", gold: 70, silver: 60, bronze: 50 },
      { country: "Australia", gold: 60, silver: 56, bronze: 40 },
      { country: "France", gold: 50, silver: 45, bronze: 35 },
      { country: "Germany", gold: 40, silver: 30, bronze: 22 },
      { country: "Italy", gold: 40, silver: 35, bronze: 37 },
      { country: "Sweden", gold: 30, silver: 25, bronze: 27 }
    ];
    this.primaryXAxis = {
      valueType: 'Category',
      title: 'Countries'
    };
    this.primaryYAxis = {
      minimum: 0, maximum: 80,
      interval: 20, title: 'Medals',
      labelFormat: '${value}K'
    };
    this.palette = ["#E94649", "#F6B53F", "#6FAAB0", "#C4C24A"];
    this.title = 'Olympic Medals';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Data point customization

The color of individual data point or data points within a range can be customized using the options below.

Point color mapping

You can bind the color for the points from [dataSource](#) for the series using [pointColorMapping](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id='chartcontainer' [chartArea]='chartArea'
[title]='title' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'>
  <e-series-collection>
    <e-series [dataSource]='chartData'
[pointColorMapping]='pointColorMapping' type='Column' xName='x' yName='y'
[cornerRadius]='cornerRadius'> </e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public cornerRadius?: Object;
  public chartArea?: Object;
  public animation?: Object;
  public pointColorMapping?: string;
  ngOnInit(): void {
    this.chartData = [
      { x: 'Jan', y: 6.96, color: "red" },
      { x: 'Feb', y: 8.9, color: "blue" },
      { x: 'Mar', y: 12, color: "orange" },
      { x: 'Apr', y: 17.5, color: "aqua" },
      { x: 'May', y: 22.1, color: "grey" }
    ];
    this.primaryXAxis = {
      valueType: 'Category', majorGridLines: { width: 0 }, title:
'Months'
    };
    this.primaryYAxis = {
      lineStyle: { width: 0 },
```

```

        majorTickLines: { width: 0 },
        minorTickLines: { width: 0 },
        labelFormat: '{value}°C',
        title: 'Temperature'
    };
    this.title = 'USA CLIMATE - WEATHER BY MONTH';
    this.chartArea = {
        border: {
            width: 0
        }
    };
    this.cornerRadius = {
        topLeft: 10, topRight: 10
    };
    this.pointColorMapping = "color";
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Range color mapping

You can differentiate data points based on their y values using [rangeColorSettings](#) in the chart.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, BarSeriesService, ColumnSeriesService,
LineSeriesService, LegendService, DataLabelService, MultiLevelLabelService,
SelectionService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id='chartcontainer'
[selectionMode]="selectionMode" [chartArea]='chartArea' [title]='title'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[legendSettings]='legendSettings'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Column' xName='x' yName='y'
name='USA' [animation]='animation' [cornerRadius]='cornerRadius'> </e-
series>
    </e-series-collection>
    <e-rangecolorsettings>

```

```

    <e-rangecolorsetting label="1°C to 10°C" start=1 end=10
[colors]="colors1"></e-rangecolorsetting>
    <e-rangecolorsetting label="11°C to 20°C" start=11 end=20
[colors]="colors2"></e-rangecolorsetting>
    <e-rangecolorsetting label="21°C to 30°C" start=21 end=30
[colors]="colors3"></e-rangecolorsetting>
    </e-rangecolorsettings>
</ejs-chart>`
}))
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    public colors1?: string[];
    public colors2?: string[];
    public colors3?: string[];
    public selectionMode?: string;
    public cornerRadius?: Object;
    public chartArea?: Object;
    public legendSettings?: Object;
    public animation?: Object;
    ngOnInit(): void {
        this.chartData = [
            { x: "Jan", y: 6.96 },
            { x: "Feb", y: 8.9 },
            { x: "Mar", y: 12 },
            { x: "Apr", y: 17.5 },
            { x: "May", y: 22.1 },
            { x: "June", y: 25 },
            { x: "July", y: 29.4 },
            { x: "Aug", y: 29.6 },
            { x: "Sep", y: 25.8 },
            { x: "Oct", y: 21.1 },
            { x: "Nov", y: 15.5 },
            { x: "Dec", y: 9.9 }
        ];
        this.primaryXAxis = {
            valueType: 'Category', majorGridLines: { width: 0 }, title:
'Months'
        };
        this.primaryYAxis = {
            lineStyle: { width: 0 },
            majorTickLines: { width: 0 },
            minorTickLines: { width: 0 },
            labelFormat: '{value}°C',
            title: 'Temperature'
        };
        this.colors1 = ['#FFFF99'];
        this.colors2 = ['#FFA500'];
        this.colors3 = ['#FF4040'];
        this.selectionMode = "Point";
        this.title = 'USA CLIMATE - WEATHER BY MONTH';
        this.chartArea = {
            border: {
                width: 0
            }
        }
    }
}

```

```

    };
    this.cornerRadius = {
      topLeft: 10, topRight: 10
    };
    this.legendSettings = {
      mode: 'Range'
    };
    this.animation = {
      enable: false
    };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Point level customization

Marker, datalabel and fill color of each data point can be customized with [pointRender](#) and [textRender](#) event.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChartModule } from '@syncfusion/ej2-angular-charts';
import { CategoryService, DateTimeService, ScrollBarService,
  ColumnSeriesService, LineSeriesService,
  ChartAnnotationService, RangeColumnSeriesService,
  StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
import { IPointRenderEventArgs } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService, ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
(pointRender)='pointRender($event)' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold'></e-series>
  </e-series-collection>
</ejs-chart>`
})

```

```

export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public pointRender(args: IPointRenderEventArgs): void {
    let seriesColor: string[] = ['#00bdae', '#404041', '#357cd2',
    '#e56590', '#f8b883',
    '#70ad47', '#dd8abd', '#7f84e8', '#7bb4eb', '#ea7a57'];
    args.fill = seriesColor[args.point.index];
  };
  ngOnInit(): void {
    this.chartData = [
      { country: "USA", gold: 50 },
      { country: "China", gold: 40 },
      { country: "Japan", gold: 70 },
      { country: "Australia", gold: 60 },
      { country: "France", gold: 50 },
      { country: "Germany", gold: 40 },
      { country: "Italy", gold: 40 },
      { country: "Sweden", gold: 30 }
    ];
    this.primaryXAxis = {
      valueType: 'Category',
      title: 'Countries'
    };
    this.primaryYAxis = {
      minimum: 0, maximum: 80,
      interval: 20, title: 'Medals'
    };
    this.title = 'Olympic Medals';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

Chart area customization

<!-- markdownlint-disable MD036 -->

Customize the chart background

<!-- markdownlint-disable MD013 -->

Using [background](#) and [border](#) properties, you can change the background color and border of the chart.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'

```

```

import { CategoryService, DateTimeService, ScrollBarService,
ColumnSeriesService, LineSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
imports: [
    ChartModule
],
providers: [ CategoryService, DateTimeService, ScrollBarService,
LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
StackingColumnSeriesService, LegendService, TooltipService,],
standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
background='skyblue' [border]='border'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold'></e-series>
    </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    public border?: Object;
    ngOnInit(): void {
        this.chartData = [
            { country: "USA", gold: 50 },
            { country: "China", gold: 40 },
            { country: "Japan", gold: 70 },
            { country: "Australia", gold: 60 },
            { country: "France", gold: 50 },
            { country: "Germany", gold: 40 },
            { country: "Italy", gold: 40 },
            { country: "Sweden", gold: 30 }
        ];
        this.primaryXAxis = {
            valueType: 'Category',
            title: 'Countries'
        };
        this.primaryYAxis = {
            minimum: 0, maximum: 80,
            interval: 20, title: 'Medals'
        };
        this.title = 'Olympic Medals';
        this.border = { width: 2, color: '#FF0000' };
    }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Chart margin

You can set margin for chart from its container through [margin](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
  ColumnSeriesService, LineSeriesService,
  ChartAnnotationService, RangeColumnSeriesService,
  StackingColumnSeriesService, LegendService, TooltipService
  } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService, ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
background='skyblue' [border]='border' [margin]='margin'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public border?: Object;
  public margin?: Object;
  ngOnInit(): void {
    this.chartData = [
      { country: "USA", gold: 50 },
      { country: "China", gold: 40 },
      { country: "Japan", gold: 70 },
      { country: "Australia", gold: 60 },
      { country: "France", gold: 50 },
      { country: "Germany", gold: 40 },
      { country: "Italy", gold: 40 },
      { country: "Sweden", gold: 30 }
    ];
  }
}
```



```

        this.primaryXAxis = {
            valueType: 'Category',
            title: 'Countries'
        };
        this.primaryYAxis = {
            minimum: 0, maximum: 80,
            interval: 20, title: 'Medals'
        };
        this.title = 'Olympic Medals';
        this.border = { width: 2, color: '#FF0000' };
        this.margin = { left: 40, right: 40, top: 40, bottom: 40 };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Chart area background

The chart area background can be customized by using the [background](#) property in the [chartArea](#).

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
    ColumnSeriesService, LineSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
    imports: [
        ChartModule
    ],
    providers: [ CategoryService, DateTimeService, ScrollBarService,
        LineSeriesService, ColumnSeriesService,
        ChartAnnotationService, RangeColumnSeriesService,
        StackingColumnSeriesService, LegendService, TooltipService, ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[title]='title' [chartArea]='chartArea'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' [border]='border'></e-series>
    </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {

```

```

public primaryXAxis?: Object;
public chartData?: Object[];
public title?: string;
public primaryYAxis?: Object;
public border?: Object;
public chartArea?: Object;
ngOnInit(): void {
    this.chartData = [
        { country: "USA", gold: 50 },
        { country: "China", gold: 40 },
        { country: "Japan", gold: 70 },
        { country: "Australia", gold: 60 },
        { country: "France", gold: 50 },
        { country: "Germany", gold: 40 },
        { country: "Italy", gold: 40 },
        { country: "Sweden", gold: 30 }
    ];
    this.primaryXAxis = {
        valueType: 'Category',
        title: 'Countries'
    };
    this.primaryYAxis = {
        minimum: 0, maximum: 80,
        interval: 20, title: 'Medals'
    };
    this.title = 'Olympic Medals';
    this.border = { width: 2, color: 'grey' };
    this.chartArea = { background: 'skyblue' };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Animation

You can customize animation for a particular series using [animation](#) property. You can enable or disable animation of the series using `enable` property, `duration` specifies the duration of an animation and `delay` allows us to start the animation at desire time.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
    ColumnSeriesService, LineSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({

```

```

imports: [
    ChartModule
],
providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService,],
standalone: true,
selector: 'app-container',
template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='country'
yName='gold' name='Gold' [border]='border' [animation]='animation'></e-
series>
    </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    public border?: Object;
    public animation?: Object;
    ngOnInit(): void {
        this.chartData = [
            { country: "USA", gold: 50 },
            { country: "China", gold: 40 },
            { country: "Japan", gold: 70 },
            { country: "Australia", gold: 60 },
            { country: "France", gold: 50 },
            { country: "Germany", gold: 40 },
            { country: "Italy", gold: 40 },
            { country: "Sweden", gold: 30 }
        ];
        this.primaryXAxis = {
            valueType: 'Category',
            title: 'Countries'
        };
        this.primaryYAxis = {
            minimum: 0, maximum: 80,
            interval: 20, title: 'Medals'
        };
        this.title = 'Olympic Medals';
        this.border = { width: 2, color: 'grey' };
        this.animation = { enable: true };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Fluid animation

Fluid animation used to animate series with updated dataSource continues animation rather than animation whole series. You can customize animation for a particular series using `[animate]` method.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, DateTimeService, ScrollBarService,
  ColumnSeriesService, LineSeriesService,
  ChartAnnotationService, RangeColumnSeriesService,
  StackingColumnSeriesService, LegendService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { ChartComponent, ILoadedEventArgs } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ CategoryService, DateTimeService, ScrollBarService,
    LineSeriesService, ColumnSeriesService,
    ChartAnnotationService, RangeColumnSeriesService,
    StackingColumnSeriesService, LegendService, TooltipService, ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart #roundcol id="column-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
(loaded)='loaded($event)'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Tiger' width='2' [cornerRadius]='radius'> </e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public border?: Object;
  public radius?: Object;
  public execute: boolean = false;
  public count: number = 0;
  @ViewChild('roundcol')
  public chart?: ChartComponent;
  public loaded(args: ILoadedEventArgs): void {
    if (this.execute) {
      return;
    }
    let columninterval = setInterval(
      () => {
        if (document.getElementById('column-container')) {
          if (this.count === 0) {
```

```

        (this.chart as ChartComponent).series[0].dataSource
= [
    { x: 'Egg', y: 206, text: 'Bangaladesh' },
    { x: 'Fish', y: 123, text: 'Bhutn' },
    { x: 'Misc', y: 48, text: 'Nepal' },
    { x: 'Tea', y: 240, text: 'Thiland' },
    { x: 'Fruits', y: 170, text: 'Malaysia' }
];
this.execute = true;
(this.chart as ChartComponent).animate();
this.count++;
} else if (this.count === 1) {
    (this.chart as ChartComponent).series[0].dataSource
= [
    { x: 'Egg', y: 86, text: 'Bangaladesh' },
    { x: 'Fish', y: 173, text: 'Bhutn' },
    { x: 'Misc', y: 188, text: 'Nepal' },
    { x: 'Tea', y: 109, text: 'Thiland' },
    { x: 'Fruits', y: 100, text: 'Malaysia' }
];
this.execute = true;
(this.chart as ChartComponent).animate();
this.count++;
} else if (this.count === 2) {
    (this.chart as ChartComponent).series[0].dataSource
= [
    { x: 'Egg', y: 156, text: 'Bangaladesh' },
    { x: 'Fish', y: 33, text: 'Bhutn' },
    { x: 'Misc', y: 260, text: 'Nepal' },
    { x: 'Tea', y: 200, text: 'Thiland' },
    { x: 'Fruits', y: 30, text: 'Malaysia' }
];
this.execute = true;
(this.chart as ChartComponent).animate();
this.count = 0;
}
} else {
    clearInterval(columninterval);
}
},
2000
);
}
ngOnInit(): void {
    this.chartData = [
        { x: 'Egg', y: 106, text: 'Bangaladesh' },
        { x: 'Fish', y: 103, text: 'Bhutn' },
        { x: 'Misc', y: 198, text: 'Nepal' },
        { x: 'Tea', y: 189, text: 'Thiland' },
        { x: 'Fruits', y: 250, text: 'Malaysia' }
    ];
    this.primaryXAxis = {
        valueType: 'Category', interval: 1, majorGridLines: { width: 0 },
        tickPosition: 'Inside',
        labelPosition: 'Inside', labelStyle: { color: '#ffffff' }
    };
    this.primaryYAxis = {

```

```

        minimum: 0, maximum: 300, interval: 50, majorGridLines: { width: 0
    },
    majorTickLines: { width: 0 }, lineStyle: { width: 0 }, labelStyle: {
color: 'transparent' }
    };
    this.title = 'Trade in Food Groups';
    this.radius = { bottomLeft: 10, bottomRight: 10, topLeft: 10,
topRight: 10 };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Chart title

Chart can be given a title using [title](#) property, to show the information about the data plotted.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, StepLineSeriesService, LegendService,
CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { TooltipService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ TooltipService, DateTimeService, StepLineSeriesService,
LegendService, CategoryService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[titleStyle]='titleStyle'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y' width=2 name='China' [marker]='marker'></e-series>
      <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y1' width=2 name='Australia' [marker]='marker'></e-series>
      <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y2' width=2 name='Japan' [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;

```

```

public marker?: Object;
public titleStyle?: Object;
ngOnInit(): void {
    this.chartData = [
        { x: new Date(1975, 0, 1), y: 16, y1: 10, y2: 4.5 },
        { x: new Date(1980, 0, 1), y: 12.5, y1: 7.5, y2: 5 },
        { x: new Date(1985, 0, 1), y: 19, y1: 11, y2: 6.5 },
        { x: new Date(1990, 0, 1), y: 14.4, y1: 7, y2: 4.4 },
        { x: new Date(1995, 0, 1), y: 11.5, y1: 8, y2: 5 },
        { x: new Date(2000, 0, 1), y: 14, y1: 6, y2: 1.5 },
        { x: new Date(2005, 0, 1), y: 10, y1: 3.5, y2: 2.5 },
        { x: new Date(2010, 0, 1), y: 16, y1: 7, y2: 3.7 }
    ];
    this.primaryXAxis = {
        title: 'Years',
        lineStyle: { width: 0 },
        labelFormat: 'y',
        intervalType: 'Years',
        valueType: 'DateTime',
        edgeLabelPlacement: 'Shift'
    };
    this.primaryYAxis = {
        title: 'Percentage (%)',
        minimum: 0, maximum: 20, interval: 2,
        labelFormat: '{value}%'
    };
    this.marker = { visible: true, width: 10, height: 10 };
    this.title = 'Unemployment Rates 1975-2010';
    this.titleStyle = {
        fontFamily: 'Arial',
        fontStyle: 'italic',
        fontWeight: 'regular',
        color: '#E27F2D',
        size: '23px'
    }
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Title position

By using the [position](#) property in [titleStyle](#), you can position the [title](#) at left, right, top or bottom of the chart. The title is positioned at the top of the chart, by default.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChartModule } from '@syncfusion/ej2-angular-charts';

```

```

import { DateTimeService, StepLineSeriesService, LegendService,
CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { TooltipService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ TooltipService, DateTimeService, StepLineSeriesService,
LegendService, CategoryService, LineSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[titleStyle]='titleStyle'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y' width=2 name='China' [marker]='marker'></e-series>
      <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y1' width=2 name='Australia' [marker]='marker'></e-series>
      <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y2' width=2 name='Japan' [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
  public titleStyle?: Object;
  ngOnInit(): void {
    this.chartData = [
      { x: new Date(1975, 0, 1), y: 16, y1: 10, y2: 4.5 },
      { x: new Date(1980, 0, 1), y: 12.5, y1: 7.5, y2: 5 },
      { x: new Date(1985, 0, 1), y: 19, y1: 11, y2: 6.5 },
      { x: new Date(1990, 0, 1), y: 14.4, y1: 7, y2: 4.4 },
      { x: new Date(1995, 0, 1), y: 11.5, y1: 8, y2: 5 },
      { x: new Date(2000, 0, 1), y: 14, y1: 6, y2: 1.5 },
      { x: new Date(2005, 0, 1), y: 10, y1: 3.5, y2: 2.5 },
      { x: new Date(2010, 0, 1), y: 16, y1: 7, y2: 3.7 }
    ];
    this.primaryXAxis = {
      title: 'Years',
      lineStyle: { width: 0 },
      labelFormat: 'y',
      intervalType: 'Years',
      valueType: 'DateTime',
      edgeLabelPlacement: 'Shift'
    };
    this.primaryYAxis = {
      title: 'Percentage (%)',
      minimum: 0, maximum: 20, interval: 2,
      labelFormat: '{value}%'
    };
    this.marker = { visible: true, width: 10, height: 10 };
  }
}

```



```

        this.title = 'Unemployment Rates 1975-2010';
        this.titleStyle = {
            position: 'Bottom'
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

The custom option helps you to position the title anywhere in the chart using [x](#) and [y](#) coordinates.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, StepLineSeriesService, LegendService,
CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { TooltipService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ TooltipService, DateTimeService, StepLineSeriesService,
LegendService, CategoryService, LineSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[titleStyle]='titleStyle'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y' width=2 name='China' [marker]='marker'></e-series>
      <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y1' width=2 name='Australia' [marker]='marker'></e-series>
      <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y2' width=2 name='Japan' [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
  public titleStyle?: Object;
  ngOnInit(): void {
    this.chartData = [
      { x: new Date(1975, 0, 1), y: 16, y1: 10, y2: 4.5 },

```

```

        { x: new Date(1980, 0, 1), y: 12.5, y1: 7.5, y2: 5 },
        { x: new Date(1985, 0, 1), y: 19, y1: 11, y2: 6.5 },
        { x: new Date(1990, 0, 1), y: 14.4, y1: 7, y2: 4.4 },
        { x: new Date(1995, 0, 1), y: 11.5, y1: 8, y2: 5 },
        { x: new Date(2000, 0, 1), y: 14, y1: 6, y2: 1.5 },
        { x: new Date(2005, 0, 1), y: 10, y1: 3.5, y2: 2.5 },
        { x: new Date(2010, 0, 1), y: 16, y1: 7, y2: 3.7 }
    ];
    this.primaryXAxis = {
        title: 'Years',
        lineStyle: { width: 0 },
        labelFormat: 'Y',
        intervalType: 'Years',
        valueType: 'DateTime',
        edgeLabelPlacement: 'Shift'
    };
    this.primaryYAxis = {
        title: 'Percentage (%)',
        minimum: 0, maximum: 20, interval: 2,
        labelFormat: '{value}%'
    };
    this.marker = { visible: true, width: 10, height: 10 };
    this.title = 'Unemployment Rates 1975-2010';
    this.titleStyle = {
        position: 'Custom',
        location: { x: 200, y: 20 }
    }
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Title alignment

You can align the title to the near, far, or center of the chart using the [textAlignment](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, StepLineSeriesService, LegendService,
CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { TooltipService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ TooltipService, DateTimeService, StepLineSeriesService,
LegendService, CategoryService, LineSeriesService],
  standalone: true,

```

```

        selector: 'app-container',
        template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[titleStyle]='titleStyle'>
            <e-series-collection>
                <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y' width=2 name='China' [marker]='marker'></e-series>
                <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y1' width=2 name='Australia' [marker]='marker'></e-series>
                <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y2' width=2 name='Japan' [marker]='marker'></e-series>
            </e-series-collection>
        </ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public primaryYAxis?: Object;
        public marker?: Object;
        public titleStyle?: Object;
        ngOnInit(): void {
            this.chartData = [
                { x: new Date(1975, 0, 1), y: 16, y1: 10, y2: 4.5 },
                { x: new Date(1980, 0, 1), y: 12.5, y1: 7.5, y2: 5 },
                { x: new Date(1985, 0, 1), y: 19, y1: 11, y2: 6.5 },
                { x: new Date(1990, 0, 1), y: 14.4, y1: 7, y2: 4.4 },
                { x: new Date(1995, 0, 1), y: 11.5, y1: 8, y2: 5 },
                { x: new Date(2000, 0, 1), y: 14, y1: 6, y2: 1.5 },
                { x: new Date(2005, 0, 1), y: 10, y1: 3.5, y2: 2.5 },
                { x: new Date(2010, 0, 1), y: 16, y1: 7, y2: 3.7 }
            ];
            this.primaryXAxis = {
                title: 'Years',
                lineStyle: { width: 0 },
                labelFormat: 'y',
                intervalType: 'Years',
                valueType: 'DateTime',
                edgeLabelPlacement: 'Shift'
            };
            this.primaryYAxis = {
                title: 'Percentage (%)',
                minimum: 0, maximum: 20, interval: 2,
                labelFormat: '{value}%'
            };
            this.marker = { visible: true, width: 10, height: 10 };
            this.title = 'Unemployment Rates 1975-2010';
            this.titleStyle = {
                position: 'Bottom',
                textAlignment: 'Far'
            }
        }
    }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Title wrap

Chart can be given a title using [title](#) property, to show the information about the data plotted.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, StepLineSeriesService, LegendService,
CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { TooltipService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ TooltipService, DateTimeService, StepLineSeriesService,
LegendService, CategoryService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[titleStyle]='titleStyle'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='month'
yName='sales' width=2 name='China' [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  public titleStyle?: Object;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = [
      { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
      { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
      { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
      { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
      { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
      { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
    ];

    this.primaryXAxis = {
      valueType: 'Category',
    };
    this.marker = { visible: true, width: 10, height: 10 };
    this.title = 'Unemployment Rates 1975-2010';
    this.titleStyle = {
```

```

        size: '18px', color: 'Red', textAlign: 'Far', textOverflow:
'Wrap'
    }
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Chart subTitle

Chart can be given a subtitle using [subTitle](#) property, to show the information about the data plotted.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, StepLineSeriesService, LegendService,
CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { TooltipService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ TooltipService, DateTimeService, StepLineSeriesService,
LegendService, CategoryService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[subTitle]='subTitle' [subTitleStyle]='subTitleStyle'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y' width=2 name='China' [marker]='marker'></e-series>
      <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y1' width=2 name='Australia' [marker]='marker'></e-series>
      <e-series [dataSource]='chartData' type='StepLine' xName='x'
yName='y2' width=2 name='Japan' [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public subTitle?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
  public subTitleStyle?: Object;
  ngOnInit(): void {
    this.chartData = [

```

```

        { x: new Date(1975, 0, 1), y: 16, y1: 10, y2: 4.5 },
        { x: new Date(1980, 0, 1), y: 12.5, y1: 7.5, y2: 5 },
        { x: new Date(1985, 0, 1), y: 19, y1: 11, y2: 6.5 },
        { x: new Date(1990, 0, 1), y: 14.4, y1: 7, y2: 4.4 },
        { x: new Date(1995, 0, 1), y: 11.5, y1: 8, y2: 5 },
        { x: new Date(2000, 0, 1), y: 14, y1: 6, y2: 1.5 },
        { x: new Date(2005, 0, 1), y: 10, y1: 3.5, y2: 2.5 },
        { x: new Date(2010, 0, 1), y: 16, y1: 7, y2: 3.7 }
    ];
    this.primaryXAxis = {
        title: 'Years',
        lineStyle: { width: 0 },
        labelFormat: 'y',
        intervalType: 'Years',
        valueType: 'DateTime',
        edgeLabelPlacement: 'Shift'
    };
    this.primaryYAxis = {
        title: 'Percentage (%)',
        minimum: 0, maximum: 20, interval: 2,
        labelFormat: '{value}%'
    };
    this.marker = { visible: true, width: 10, height: 10 };
    this.title = 'Unemployment Rates 1975-2010';
    this.subTitle = '(1975-2010)';
    this.subTitleStyle = {
        fontFamily: 'Arial',
        fontStyle: 'italic',
        fontWeight: 'regular',
        color: '#E27F2D',
        size: '20px'
    }
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See also

- [Customize the series points using patterns](#)

<!-- markdownlint-disable MD036 -->

Render methods in Angular Chart component

Chart uses following two rendering methods.

- SVG
- Canvas

SVG

SVG is used to render Chart by default for all browsers except IE8 and old versions.

Canvas

You can switch between SVG and Canvas rendering by using the `enableCanvas` option. The canvas mode rendering is used in the following scenarios,

- Plotting large number of data points.
- Performing high frequency live updates.

Limitations

- Animation is not supported.

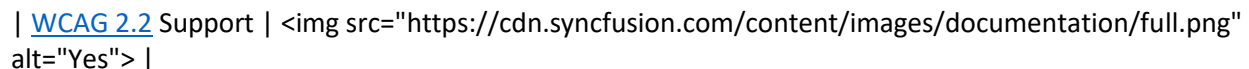
Accessibility in Angular Chart component

The Chart component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Chart component is outlined below.

| Accessibility Criteria | Compatibility |

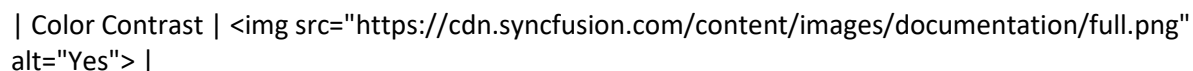
| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes"> |

| [Section 508](#) Support |  alt="Yes"> |

| Screen Reader Support |  alt="Yes"> |

| Right-To-Left Support |  alt="Yes"> |

| Color Contrast |  alt="Yes"> |

| Mobile Device Support |  alt="Yes"> |

| Keyboard Navigation Support |  alt="Yes"> |

| [Accessibility Checker](#) Validation |  alt="Yes"> |

| [Axe-core](#) Accessibility Validation |  alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

```
margin: 0.5em 0;
```

```
}
```

```
</style>
```

```
<div> - All
features of the component meet the requirement.</div>
```

```
<div> - Some features of the component do not meet the requirement.</div>
```

```
<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Chart component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Chart component:

- `img` (role)
- `button` (role)
- `region` (role)
- `aria-label` (attribute)
- `aria-hidden` (attribute)
- `aria-pressed` (attribute)

Keyboard interaction

The Chart component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Chart component.

| Press | To do this |

| --- | --- |

| Alt + J | Moves the focus to the chart element. |

| Tab | Moves the focus to the next element in the chart. |

| Shift + Tab | Moves the focus to the previous element in the chart. |

| Down Arrow | Moves the focus to the data point left side from the selected point. |

| Up Arrow | Moves the focus to the data point right side from the selected point. |

| Left Arrow | Moves the focus to the next series in the chart. |

| Right Arrow | Moves the focus to the previous series in the chart. |

| ESC | Cancel the tooltip for the data point. |

| Enter/Space | Selects the data point in the series. |

| Down/Left Arrow | Moves the focus to the legend left side from the selected legend. |

| Up/Right Arrow | Moves the focus to the legend right side from the selected legend. |

| Enter/Space | Toggles the visibility of the corresponding series. |

| Ctrl + + | Zoom in the chart. |

| Ctrl + - | Zoom out the chart. |

| Down/Up Arrow | Pan the chart vertically. |

| Left/Right Arrow | Pan the chart horizontally. |

| R | Reset the zoomed chart. |

| Ctrl + P | Prints the Chart. |

Ensuring accessibility

The Chart component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Chart component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Chart component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Internationalization in Angular Chart component

Chart provide supports for internationalization for below chart elements.

- Datalabel.
- Axis label.
- Tooltip.

For more information about number and date formatter you can refer [internationalization](#).

<!-- markdownlint-disable MD036 -->

Globalization

Globalization is the process of designing and developing an component that works in different cultures/locales. Internationalization library is used to globalize number, date, time values in Chart component using `labelFormat` property in axis.

Numeric Format

In the below example axis, point and tooltip labels are globalized to EUR.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { LegendService, TooltipService, DataLabelService,
ColumnSeriesService, DateTimeService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { setCurrencyCode } from '@syncfusion/ej2-base';
setCurrencyCode('EUR');
@Component({
```

```

imports: [
    ChartModule
],
providers: [ LegendService, TooltipService, DataLabelService,
ColumnSeriesService, DateTimeService],
standalone: true,
selector: 'app-container',
template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[title]='title' [tooltip]='tooltip'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y' name='Product X' [marker]='marker'></e-series>
        <e-series [dataSource]='chartData' type='Column' xName='x'
yName='y1' name='Product Y' [marker]='marker'></e-series>
    </e-series-collection>
</ejs-chart>`
))
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public tooltip?: Object;
    public marker?: Object;
    public primaryYAxis?: Object;
    ngOnInit(): void {
        this.chartData = [
            { x: 1900, y: 4, y1: 2.6, y2: 2.8 }, { x: 1920, y: 3.0, y1:
2.8, y2: 2.5 },
            { x: 1940, y: 3.8, y1: 2.6, y2: 2.8 }, { x: 1960, y: 3.4,
y1: 3, y2: 3.2 },
            { x: 1980, y: 3.2, y1: 3.6, y2: 2.9 }, { x: 2000, y: 3.9,
y1: 3, y2: 2 }
        ];
        this.primaryXAxis = {
            title: 'Year',
            edgeLabelPlacement: 'Shift'
        };
        this.primaryYAxis = {
            title: 'Sales Amount in Millions',
            labelFormat: 'c'
        };
        this.tooltip = {
            enable: true, format: '${series.name} <br>${point.x} :
${point.y}'
        };
        this.marker = {
            dataLabel: {
                visible: true
            }
        };
        this.title = 'Average Sales Comparison';
    }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Localization in Angular Chart component

Localization library allows to localize the default text content of Chart. In Chart component, it has the static text on some features(like zooming toolbars) and this can be changed to any other culture(Arabic, Deutsch, French, etc) by defining the locale value and translation object.

<!-- markdownlint-disable MD033 -->

Locale key words	Text to display
Zoom	Zoom
ZoomIn	ZoomIn
ZoomOut	ZoomOut
Reset	Reset
Pan	Pan
ResetZoom	Reset Zoom

To load translation object in an application use load function of L10n class.

For more information about localization, refer this [localization](#)

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { DateTimeService, AreaSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, ZoomService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { L10n } from '@syncfusion/ej2-base';
L10n.load({
  'ar-AR': {
    'chart': {
      ZoomIn: 'تكبير',
      ZoomOut: 'تصغير',
      Zoom: 'زوم',
      Pan: 'مقلاة',
      Reset: 'إعادة تعيين',
    },
  },
});
@Component({
  imports: [
    ChartModule
  ],
```

```

providers: [ DateTimeService, AreaSeriesService, LegendService,
ZoomService],
standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
locale='ar-AR' [zoomSettings]='zoom'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Area' xName='x'
yName='y' name='Product X' ></e-series>
      <e-series [dataSource]='chartData' type='Area' xName='x'
yName='y1' name='Product Y'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;

  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  public primaryYAxis?: Object;
  public zoom?: Object;
  ngOnInit(): void {
    this.chartData = [
      { x: 1900, y: 4, y1: 2.6, y2: 2.8 }, { x: 1920, y: 3.0, y1:
2.8, y2: 2.5 },
      { x: 1940, y: 3.8, y1: 2.6, y2: 2.8 }, { x: 1960, y: 3.4,
y1: 3, y2: 3.2 },
      { x: 1980, y: 3.2, y1: 3.6, y2: 2.9 }, { x: 2000, y: 3.9,
y1: 3, y2: 2 }
    ];
    this.primaryXAxis = {
      title: 'Year',
      edgeLabelPlacement: 'Shift'
    };
    this.primaryYAxis = {
      title: 'Sales Amount in Millions',
    };
    this.zoom = {
      enableMouseWheelZooming: true,
      enableDeferredZooming: true,
      enablePinchZooming: true,
      enableSelectionZooming: true
    }
    this.title = 'Average Sales Comparison';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD033 -->

<!-- markdownlint-disable MD038 -->

Ej1 api migration in Angular Chart component

This article describes the API migration process of {Component Name} component from Essential JS 1 to Essential JS 2.

Chart

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Annotation | **Property:** *annotation*

 <ej-chart>
<e-annotations>
<e-annotation>
<e-annotation>
</e-annotations>
</ej-chart> | **Property:** *e-annotation*

 <ejs-chart>
<e-annotations>
<e-annotation>
<e-annotation>
</e-annotations>
</ejs-chart> |

| Background | **Property:** *background*

 <ej-chart background='red'>
</ej-chart> | **Property:** *background*

 <ej-chart background='red'>
</ej-chart> |

| backgroundImageUrl | **Property:** *backgroundImageUrl*

 <ej-chart background='red'>
</ej-chart> | **Property:** Not Applicable.

| border of the chart | **Property:** *border*

 <ej-chart [border]='border'>
</ej-chart>

<code>this.border = { width: 2, color: 'red' } | **Property:** *border*

 <ejs-chart [isResponsive]='border'>
</ejs-chart>

<code>this.border = { width: 2, color: 'red' } |

| isResponsive of chart | **Property:** *isResponsive*

 <ej-chart [isResponsive]='response'>
</ej-chart>

<code>this.response = true | **Property:** Not Applicable |

| Chart area of the chart | **Property:** *chartArea*

 <ej-chart [chartArea]='chartArea'>
</ej-chart>

<code>this.chartArea = { border: { color: 'red', width: 2 }, background: 'transparent' } | **Property:** *chartArea*

 <ejs-chart [chartArea]='chartArea'>
</ejs-chart>

<code>this.chartArea = { border: { color: 'red', width: 2 }, background: 'transparent' } |

| Column of the chart | **Property:** *columnDefintions*

 <ej-chart>
<e-columnDefinitions>
<e-columnDefinition>
</e-columnDefinition>
</e-columnDefinitions>
</ej-chart> | **Property:** *columns*

 <ej-chart>
<e-columns>
<e-column>
</e-column>
</e-columns>
</ej-chart> |

| crossHair | **Property:** *crossHair*

 <ej-chart [crosshair]='crosshair'>
</ej-chart>

<code>this.crosshair = { } | **Property:** *crossHair*

 <ejs-chart [crosshair]='crosshair'>
</ejs-chart>

<code>this.crosshair = { } |

| Common series options | **Property:** *commonSeriesOptions*

 <ej-chart [commonSeriesOptions]='series'>
</ej-chart>

<code>this.series = { } | **Property:** Not Applicable

| Indicators | **Property:** *indicator*

 <ej-chart>
<e-indicators>
<e-indicator>
</e-indicator>
</e-indicators>
</ej-chart> | **Property:** *indicator*

 <ejs-chart>
<e-indicators>
<e-indicator>
</e-indicator>
</e-indicators>
</ejs-chart> |

| Rows of the chart | **Property:** *rowDefintions*

 <ej-chart>
 <e-rowDefinitions>
 <e-rowDefinition>
 </e-rowDefinition>
 </e-rowDefinitions>
 </ej-chart> | **Property:** *rows*

 <ej-chart>
 <e-columns>
 <e-column>
 </e-column>
 </e-columns>
 </ej-chart>

| primaryXAxis | **Property:** *primaryXAxis*

 <ej-chart [primaryXAxis]='primaryXAxis'>
 </ej-chart>

 <code>this.primaryXAxis = { } | **Property:** *primaryXAxis*

 <ejs-chart [primaryXAxis]='primaryXAxis'>
 </ejs-chart>

 <code>this.primaryXAxis = { } |

| primaryYAxis | **Property:** *primaryYAxis*

 <ej-chart [primaryYAxis]='primaryYAxis'>
 </ej-chart>

 <code>this.series = { } | **Property:** *primaryXAxis*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
 </ejs-chart>

 <code>this.primaryYAxis = { } |

| Series | **Property:** *series*

 <ej-chart>
 <e-series-collection>
 <e-series>
 </e-series>
 </e-series-collection>
 </ej-chart> | **Property:** *series*

 <ejs-chart>
 <e-series-collection>
 <e-series>
 </e-series>
 </e-series-collection>
 </ejs-chart> |

| Selected data | **Property:** *selectedDataPointIndexes*

 <ej-chart [selectedDataPointIndexes]='selectData'>
 </ej-chart>

 <code>this.selectedData = [{ seriesIndex: 2, pointIndex: 2}] | **Property:** *primaryXAxis*

 <ejs-chart [selectedDataIndexes]='selectData'>
 </ejs-chart>

 <code>this.selectData = [{ series: 0, point: 1}] |

| Chart theme | **Property:** *theme*

 <ej-chart [theme]='theme'>
 </ej-chart>

 <code>this.theme = 'Material' | **Property:** *theme*

 <ejs-chart [theme]='theme'>
 </ejs-chart>

 <code>this.theme = 'Fabric' |

| Side by side placement of series | **Property:** *sideBySideSeriePlacement*

 <ej-chart [sideBySideSeriePlacement]='sideBySideSeriePlacement'>
 </ej-chart>

 <code>this.sideBySideSeriePlacement = true | **Property:** *enableSideBySidePlacement*

 <ejs-chart [enableSideBySidePlacement]='sideBySide'>
 </ejs-chart>

 <code>this.sideBySide = true |

| Chart title | **Property:** *title.text*

 <ej-chart [title]='title'>
 </ej-chart>

 <code>this.title = 'Chart title' | **Property:** *title*

 <ejs-chart [title]='sideBySide'>
 </ejs-chart>

 <code>this.title = 'Chart title' |

| Zoom settings | **Property:** *zooming*

 <ej-chart [zooming]='zoom'>
 </ej-chart>

 <code>this.zoom = {enable : true, enablePinching : true, enableMouseWheel: true, enableScrollBar: true, enableDeferredZoom: true, toolBarItems: [], type: 'Y'} | **Property:** *zoomSettings*

 <ejs-chart [zoomSettings]='zoom'>
 </ejs-chart>

 <code>this.zoom = {enablePinchZooming: true, enableMouseWheelZooming: true, enableDeferredZooming: true, enablePinchZooming: true, enableSelectionZooming: true, enablePan: true, enableScrollBar: true } |

| Multi selection of chart | Not Applicable | **Property:** *isMultiSelect*

 <ejs-chart [isMultiSelect]='select'>
 </ejs-chart>

 <code>this.select = true |

| Vertical chart | **Property:** *isTransposed*

 <ej-chart>
 <e-series-collection>
 <e-series [isTransposed]='isTransposed'>
 </e-series>
 </e-series-collection>
 </ej-chart>

`<code>this.isTransposed = true` | **Property:** *isTransposed* `</code>` `<ej-chart [isTransposed]='isTransposed'>` `</ej-chart>` `<code>this.isTransposed = true` |

| Tooltip for chart | **Property:** *tooltip* `</code>` `<ej-chart>` `<e-series-collection>` `<e-series [tooltip]='tooltip'>` `</e-series>` `</e-series-collection>` `</ej-chart>` `
` `<code>this.tooltip = {}` | **Property:** *isTransposed* `</code>` `<ej-chart [tooltip]='tooltip'>` `</ej-chart>` `
` `<code>this.tooltip = {}` |

| Height of chart | **Property:** *size.height* `</code>` `<ej-chart [size]='size'>` `</ej-chart>` `
` `<code>this.size = { height: '300'}` | **Property:** *height* `</code>` `<ej-chart [height]='height'>` `</ej-chart>` `
` `<code>this.height = '300'` |

| Width of chart | **Property:** *size.width* `</code>` `<ej-chart [size]='size'>` `</ej-chart>` `
` `<code>this.size = { width: '300'}` | **Property:** *height* `</code>` `<ej-chart [width]='width'>` `</ej-chart>` `
` `<code>this.width = '300'` |

3DChart

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| 3D chart | **Property:** *enable3D* `</code>` `<ej-chart [enable3D]='enable'>` `</ej-chart>` `
` `<code>this.enable = true` | **Property:** Not Applicable |

Annotations

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Annotation | **Property:** *annotation* `</code>` `<ej-chart>` `<e-annotations>` `<e-annotation [visible]='visible'>` `</e-annotation>` `</e-annotations>` `</ej-chart>` `
` `<code>this.visible = true` | **Property:** *e-annotation* `</code>` `<ej-chart>` `<e-annotations>` `<e-annotation>` `</e-annotations>` `</ej-chart>` |

| Angle of annotation | **Property:** *annotation.angle* `</code>` `<ej-chart>` `<e-annotations>` `<e-annotation [angle]='angle'>` `</e-annotation>` `</e-annotations>` `</ej-chart>` `
` `<code> this.angle = 60` | **Property:** Not Applicable

| Annotation content | **Property:** *annotation.content* `</code>` `<ej-chart>` `<e-annotations>` `<e-annotation>` `<e-annotation content='<div>Chart</div>'>` `</e-annotation>` `</e-annotations>` `</ej-chart>` | **Property:** *annotation.content* `</code>` `<ej-chart>` `<e-annotations>` `<e-annotation>` `<e-annotation content='<div>Chart</div>'>` `</e-annotation>` `</e-annotations>` `</ej-chart>` |

| Coordinate unit of annotation | **Property:** *annotation.coordinateUnit* `</code>` `<ej-chart>` `<e-annotations>` `<e-annotation>` `<e-annotation coordinateUnit='Pixel'>` `</e-annotation>` `</e-annotations>` `</ej-chart>` | **Property:** *annotation.coordinateUnits* `</code>` `<ej-chart>` `<e-annotations>` `<e-annotation>` `<e-annotation coordinateUnits='Pixel'>` `</e-annotation>` `</e-annotations>` `</ej-chart>` |

| Horizontal alignment of annotation | **Property:** *annotation.horiontalAlignment* `</code>` `<ej-chart>` `<e-annotations>` `<e-annotation>` `<e-annotation`

horizontalAlignment='near'></e-annotations></ej-chart> | **Property:** *annotation.horizontalAlignment*

 <ejs-chart>
<e-annotations>
<e-annotation horizontalAlignment='Near'></e-annotations>
</ejs-chart>|

| Margin for annotation | **Property:** *annotation.margin*

 <ej-chart>
<e-annotations>
<e-annotation>
<e-annotation [margin]='margin'></e-annotations>
</ej-chart>
<code>this.margin = { } | **Property:** Not Applicable.

| Opacity for annotation | **Property:** *annotation.opacity*

 <ej-chart>
<e-annotations>
<e-annotation>
<e-annotation [opacity]='opacity'></e-annotations>
</ej-chart>
<code>this.opacity = 2 | **Property:** Not Applicable.

| region of annotation | **Property:** *annotation.region*

 <ej-chart>
<e-annotations>
<e-annotation>
<e-annotation region='Chart'></e-annotations>
</ej-chart> | **Property:** *annotation.region*

 <ejs-chart>
<e-annotations>
<e-annotation>
<e-annotation region='Chart'></e-annotations>
</ejs-chart>|

| Vertical alignment of annotation | **Property:** *annotation.verticalAlignment*

 <ej-chart>
<e-annotations>
<e-annotation>
<e-annotation verticalAlignment='Top'></e-annotations>
</ej-chart> | **Property:** *annotation.verticalAlignment*

 <ejs-chart>
<e-annotations>
<e-annotation>
<e-annotation verticalAlignment='Top'></e-annotations>
</ejs-chart>|

| XValue of annotation | **Property:** *annotation.x*

 <ej-chart>
<e-annotations>
<e-annotation>
<e-annotation [x]='xvalue'></e-annotations>
</ej-chart>

<code> this.xvalue = 2 | **Property:** *annotation.x*

 <ejs-chart>
<e-annotations>
<e-annotation>
<e-annotation x='xvalue'></e-annotations>
</ejs-chart>

<code>this.xvalue = 2 |

| x axis name of annotation | **Property:** *annotation.xAxisName*

 <ej-chart>
<e-annotations>
<e-annotation>
<e-annotation [xAxisName]='xvalue'></e-annotations>
</ej-chart>
<code> this.xvalue = 'axis' | **Property:** *annotation.xAxisName*

 <ejs-chart>
<e-annotations>
<e-annotation>
<e-annotation [xAxisName]='xvalue'></e-annotations>
</ejs-chart>

<code>this.xvalue = 'axis' |

| y axis name of annotation | **Property:** *annotation.yAxisName*

 <ej-chart>
<e-annotations>
<e-annotation>
<e-annotation [yAxisName]='yvalue'></e-annotations>
</ej-chart>

<code> this.yvalue = 'axis' | **Property:** *annotation.xAxisName*

 <ejs-chart>
<e-annotations>
<e-annotation>
<e-annotation [yAxisName]='yvalue'></e-annotations>
</ejs-chart>

<code>this.yvalue = 'axis' |

Columns

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Column of the chart | **Property:** *columnDefintions*

 <ej-chart>
<e-columnDefinitions>
<e-columnDefinition>
</e-columnDefinition>
</e-

columnDefinitions>
</ej-chart> | **Property:** *columns*

<ej-chart>
<e-columns>
</e-column>
</e-column>
</e-columns>
</ej-chart>

| Width of columns | **Property:** *columnDefinition.width*

<ej-chart>
<e-columnDefinitions>
<e-columnDefinition [width]='width'>
</e-columnDefinition>
</e-columnDefinitions>
</ej-chart>this.width = 20;

<code> | **Property:** *column.width*

<ej-chart>
<e-columns>
<e-column [width]='width'>
</e-column>
</e-columns>
</ej-chart>

 this.width = '400';

| Unit of column width | **Property:** *column.unit*

<ej-chart>
<e-columnDefinitions>
<e-columnDefinition [unit]='unit'>
</e-columnDefinition>
</e-columnDefinitions>
</ej-chart>this.unit = 'Pixel';

<code> | **Property:** Not Applicable;

| Line color of columns | **Property:** *columnDefintions.lineColor*

<ej-chart>
<e-columnDefinitions>
<e-columnDefinition [lineColor]='color'>
</e-columnDefinition>
</e-columnDefinitions>
</ej-chart>this.color = 'blue';

<code> | **Property:** *column.border.color*

<ej-chart>
<e-columns>
<e-column [border]='border'>
</e-column>
</e-columns>
</ej-chart>

 this.border = { color: 'red' }

| Line width of columns | **Property:** *columnDefintions.lineWidth*

<ej-chart>
<e-columnDefinitions>
<e-columnDefinition [lineWidth]='width'>
</e-columnDefinition>
</e-columnDefinitions>
</ej-chart>this.width = 2;

<code> | **Property:** *column.border.width*

<ej-chart>
<e-columns>
<e-column [border]='border'>
</e-column>
</e-columns>
</ej-chart>

 this.border = { width: 2 }

Rows

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| row of the chart | **Property:** *rowDefintions*

<ej-chart>
<e-rowDefinitions>
<e-rowDefinition>
</e-rowDefinition>
</e-rowDefinitions>
</ej-chart> | **Property:** *rows*

<ej-chart>
<e-rows>
<e-row>
</e-row>
</e-rows>
</ej-chart>

| height of rows | **Property:** *rowDefinition.height*

<ej-chart>
<e-rowDefinitions>
<e-rowDefinition [height]='height'>
</e-rowDefinition>
</e-rowDefinitions>
</ej-chart>this.height = 20;

<code> | **Property:** *row.height*

<ej-chart>
<e-rows>
<e-row [height]='height'>
</e-row>
</e-rows>
</ej-chart>

 this.height = '400';

| Unit of row height | **Property:** *row.unit*

<ej-chart>
<e-rowDefinitions>
<e-rowDefinition [unit]='unit'>
</e-rowDefinition>
</e-rowDefinitions>
</ej-chart>this.unit = 'Pixel';

<code> | **Property:** Not Applicable;

| Line color of rows | **Property:** *rowDefintions.lineColor*

<ej-chart>
<e-rowDefinitions>
<e-rowDefinition [lineColor]='color'>
</e-rowDefinition>
</e-rowDefinitions>
</ej-chart>this.color = 'blue';

<code> | **Property:** *row.border.color*

```
<br/> <br/><ej-chart> <br><e-rows><br><e-row [border]='border'><br></e-row><br></e-rows><br></ej-chart> <br> <br> this.border = { color: 'red' }
```

| Line height of rows | **Property:** *rowDefinitions.lineheight*

 <ej-chart>
<e-rowDefinitions>
<e-rowDefinition [lineheight]='height'>
</e-rowDefinition>
</e-rowDefinitions>
</ej-chart>this.height = 2;

 <code> | **Property:** *row.border.height*

<ej-chart>
<e-rows>
<e-row [border]='border'>
</e-row>
</e-rows>
</ej-chart>

 this.border = { width: 2 }

Common Series Options

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Common series option | **Property:** *commonSeriesOptions*

 <ej-chart
[commonSeriesOptions]='commonSeriesOption'>
</ej-chart>

<code>this.commonSeriesOption = { } | **Property:** *border*

 <ejs-chart
[commonSeriesOption]='commonSeriesOption'>
</ejs-chart>

<code>this.commonSeriesOptions = { }

Crosshair

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| crossHair | **Property:** *visible*

 <ej-chart [crosshair]='crosshair'>
</ej-chart>

<code>this.crosshair = { visible: true } | **Property:** *enable*

 <ejs-chart
[crosshair]='crosshair'>
</ejs-chart>

<code>this.crosshair = { enable: true }

| type of cross hair | **Property:** *type*

 <ej-chart [crosshair]='crosshair'>
</ej-chart>

<code>this.crosshair = { visible: true } | **Property:** Not Applicable

| Trackball settings of crosshair | **Property:** *trackBallSettings*

 <ej-chart
[crosshair]='crosshair'>
</ej-chart>

<code>this.crosshair = { trackBallSettings: { } } |
Property: Not Applicable.

| Marker settings of crosshair | **Property:** *markerSettings*

 <ej-chart
[crosshair]='crosshair'>
</ej-chart>

<code>this.crosshair = { markerSettings: { } } |
Property: Not Applicable

| crossHair line style | **Property:** *line*

 <ej-chart [crosshair]='crosshair'>
</ej-chart>

<code>this.crosshair = { line: { color: 'red', width: 2 } } | **Property:** *line*

 <ejs-chart
[crosshair]='crosshair'>
</ejs-chart>

<code>this.crosshair = { line :{ color: 'red', width: 2
} }

Indicator

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Indicators | **Property:** *indicator*

 <ej-chart>
<e-indicators>
<e-
indicator>
</e-indicator>
</e-indicators>
</ej-chart> | **Property:** *indicator*


```
</><ejs-chart> <br><e-indicators><br><e-indicator><br></e-indicator><br></e-
indicators><br></ejs-chart>
```

```
| Indicator type | Property: type <br/> <br/> <ej-chart> <br><e-indicators><br><e-indicator
type='Tma'><br></e-indicator><br></e-indicators><br></ej-chart> | Property: type <br/>
<br/><ejs-chart> <br><e-indicators><br><e-indicator type='Tma' ><br></e-indicator><br></e-
indicators><br></ejs-chart>
```

```
| Period of indicator | Property: period <br/> <br/> <ej-chart> <br><e-indicators><br><e-indicator
[period]='period'><br></e-indicator><br></e-indicators><br></ej-chart> <br> <br> <code>
this.period = 20; | Property: period <br/> <br/><ejs-chart> <br><e-indicators><br><e-indicator
[period]='period' ><br></e-indicator><br></e-indicators><br></ejs-chart> <br> <br> <code>
this.period = 14;
```

```
[%d value of indicator | Property: dPeriod <br/> <br/> <ej-chart> <br><e-indicators><br><e-
indicator [dPeriod]='period'><br></e-indicator><br></e-indicators><br></ej-chart> <br> <br>
<code> this.dperiod = 20; | Property: dPeriod <br/> <br/><ejs-chart> <br><e-indicators><br><e-
indicator [dPeriod]='period' ><br></e-indicator><br></e-indicators><br></ejs-chart> <br> <br>
<code> this.dperiod = 14;
```

```
[%k value of indicator | Property: kPeriod <br/> <br/> <ej-chart> <br><e-indicators><br><e-
indicator [kPeriod]='period'><br></e-indicator><br></e-indicators><br></ej-chart> <br> <br>
<code> this.dperiod = 20; | Property: kPeriod <br/> <br/><ejs-chart> <br><e-indicators><br><e-
indicator [kPeriod]='period' ><br></e-indicator><br></e-indicators><br></ejs-chart> <br> <br>
<code> this.dperiod = 14;
```

```
| Show overBought/overSold values of indicator | Property: Not Applicable | Property: showZones
<br/> <br/><ejs-chart> <br><e-indicators><br><e-indicator [showZones]='showZone' ><br></e-
indicator><br></e-indicators><br></ejs-chart> <br> <br> <code> this.showZone = true;
```

```
| overBought values of indicator | Property: Not Applicable | Property: overBought <br/> <br/><ejs-
chart> <br><e-indicators><br><e-indicator [overBought]='overBought' ><br></e-
indicator><br></e-indicators><br></ejs-chart> <br> <br> <code> this.overBought = 10;
```

```
| overSold values of indicator | Property: Not Applicable | Property: overSold <br/> <br/><ejs-chart>
<br><e-indicators><br><e-indicator [overBought]='overSold' ><br></e-indicator><br></e-
indicators><br></ejs-chart> <br> <br> <code> this.overSold = 10;
```

```
| Field value of indicator | Property: field <br/> <br/> <ej-chart> <br><e-indicators><br><e-
indicator [field]='field'><br></e-indicator><br></e-indicators><br></ej-chart> <br> <br> <code>
this.field = 'Close'; | Property: field <br/> <br/><ejs-chart> <br><e-indicators><br><e-indicator
[field]='field' ><br></e-indicator><br></e-indicators><br></ejs-chart> <br> <br> <code> this.field =
'Close';
```

```
| Standard deviations of indicator | Property: standardDeviation <br/> <br/> <ej-chart> <br><e-
indicators><br><e-indicator [standardDeviation]='standardDeviation'><br></e-
indicator><br></e-indicators><br></ej-chart> <br> <br> <code> this.standardDeviation = 2; |
Property: standardDeviation <br/> <br/><ejs-chart> <br><e-indicators><br><e-indicator
```

[standardDeviation]='standardDeviation' >
</e-indicator>
</e-indicators>
</ejs-chart>

 <code> this.standardDeviation = 2;

| Slow period of MACD indicator | **Property:** *shortPeriod*

 <ej-chart>
<e-indicators>
<e-indicator [shortPeriod]='period'>
</e-indicator>
</e-indicators>
</ej-chart>

 <code> this.period = 2; | **Property:** *slowPeriod*

<ejs-chart>
<e-indicators>
<e-indicator [slowPeriod]='period' >
</e-indicator>
</e-indicators>
</ejs-chart>

 <code> this.period = 2;

| Fast period of MACD indicator | **Property:** *longPeriod*

 <ej-chart>
<e-indicators>
<e-indicator [longPeriod]='period'>
</e-indicator>
</e-indicators>
</ej-chart>

 <code> this.period = 2; | **Property:** *fastPeriod*

<ejs-chart>
<e-indicators>
<e-indicator [fastPeriod]='period' >
</e-indicator>
</e-indicators>
</ejs-chart>

 <code> this.period = 2;

| Line style of MACD indicator | **Property:** *macdLine*

 <ej-chart>
<e-indicators>
<e-indicator [macdLine]='line'>
</e-indicator>
</e-indicators>
</ej-chart>

 <code> this.line = { width: 2, color: 'red'}; | **Property:** *macdLine*

<ejs-chart>
<e-indicators>
<e-indicator [macdLine]='line' >
</e-indicator>
</e-indicators>
</ejs-chart>

 <code> this.line = { color: 'red', width: 3 };

| Macd type of indicator | **Property:** *macdType*

 <ej-chart>
<e-indicators>
<e-indicator [macdType]='type'>
</e-indicator>
</e-indicators>
</ej-chart>

 <code> this.type = 'both'; | **Property:** *macdType*

<ejs-chart>
<e-indicators>
<e-indicator [macdType]='type' >
</e-indicator>
</e-indicators>
</ejs-chart>

 <code> this.type = 'both'

| Macd positive color of indicator | **Property:** Not Applicable | **Property:** *macdType*

<ejs-chart>
<e-indicators>
<e-indicator macdPositiveColor='red' >
</e-indicator>
</e-indicators>
</ejs-chart>

| Macd negative color of indicator | **Property:** Not Applicable | **Property:** *macdType*

<ejs-chart>
<e-indicators>
<e-indicator macdPositiveColor='red' >
</e-indicator>
</e-indicators>
</ejs-chart>

| Bollinger band color of indicator | **Property:** Not Applicable | **Property:** *macdType*

<ejs-chart>
<e-indicators>
<e-indicator bandColor='red' >
</e-indicator>
</e-indicators>
</ejs-chart>

| Appearance of lower line in indicator | **Property:** *lowerLine*

 <ej-chart>
<e-indicators>
<e-indicator [lowerLine]='line'>
</e-indicator>
</e-indicators>
</ej-chart>

 <code> this.line = { fill: 'red', width: 2}; | **Property:** *lowerLine*

<ejs-chart>
<e-indicators>
<e-indicator [lowerLine]='line' >
</e-indicator>
</e-indicators>
</ejs-chart>

 <code> this.line = { color: 'red', width: 2, dashArray: '10, 5', type: 'smooth'}

| Appearance of upper line in indicator | **Property:** *upperLine*

 <ej-chart>
<e-indicators>
<e-indicator [upperLine]='line'>
</e-indicator>
</e-indicators>
</ej-chart>

 <code> this.line = { fill: 'red', width: 2}; | **Property:** *upperLine*

<ejs-chart>

chart>
<e-indicators>
<e-indicator [upperLine]='line' >
</e-indicator>
</e-indicators>
</ejs-chart>

 <code> this.line = { color: 'red', width: 2, dashArray: '10, 5', type: 'smooth' }

| Appearance of period line in indicator | **Property:** *periodLine*

 <ej-chart>
<e-indicators>
<e-indicator [periodLine]='line'>
</e-indicator>
</e-indicators>
</ej-chart>

 <code> this.line = { fill: 'red', width: 2; } | **Property:** *periodLine*

<ejs-chart>
<e-indicators>
<e-indicator [periodLine]='line' >
</e-indicator>
</e-indicators>
</ejs-chart>

 <code> this.line = { color: 'red', width: 2, dashArray: '10, 5', type: 'smooth' }

| Name of the series for which indicator has to draw | **Property:** *seriesName*

 <ej-chart>
<e-indicators>
<e-indicator seriesName='series1'>
</e-indicator>
</e-indicators>
</ej-chart> | **Property:** *type*

<ejs-chart>
<e-indicators>
<e-indicator seriesName='series1' >
</e-indicator>
</e-indicators>
</ejs-chart>

| Histogram customization | **Property:** *histogram*

 <ej-chart>
<e-indicators>
<e-indicator [histogram]='series1'>
</e-indicator>
</e-indicators>
</ej-chart>

 <code> this.series1 = { } | **Property:** Not Applicable

| Enable animation for indicator | **Property:** *enableAnimation*

 <ej-chart>
<e-indicators>
<e-indicator [enableAnimation]='animation'>
</e-indicator>
</e-indicators>
</ej-chart>

 <code> this.animation = true | **Property:** *animation.enable*

<ejs-chart>
<e-indicators>
<e-indicator [animation]='animation' >
</e-indicator>
</e-indicators>
</ejs-chart>

 <code> this.animation = { enable: true }

| Animation duration for indicator | **Property:** *animationDuration*

 <ej-chart>
<e-indicators>
<e-indicator [animationDuration]='animation'>
</e-indicator>
</e-indicators>
</ej-chart>

 <code> this.animation = 10 | **Property:** *animation.duration*

<ejs-chart>
<e-indicators>
<e-indicator [animation]='animation' >
</e-indicator>
</e-indicators>
</ejs-chart>

 <code> this.animation = { duration: 1000 }

| Animation delay for indicator | **Property:** Not Applicable | **Property:** *animation.duration*

<ejs-chart>
<e-indicators>
<e-indicator [animation]='animation' >
</e-indicator>
</e-indicators>
</ejs-chart>

 <code> this.animation = { delay: 100 }

| Tooltip for indicator | **Property:** *tooltip*

 <ej-chart>
<e-indicators>
<e-indicator [tooltip]='tooltip'>
</e-indicator>
</e-indicators>
</ej-chart>

 <code> this.tooltip = { enable: true } | **Property:** *enableTooltip*

<ejs-chart>
<e-indicators>
<e-indicator [enableTooltip]='tooltip' >
</e-indicator>
</e-indicators>
</ejs-chart>

 <code> this.tooltip = true;

| Trigger value for indicator | **Property:** *trigger*

 <ej-chart>
<e-indicators>
<e-indicator [trigger]='trigger'>
</e-indicator>
</e-indicators>
</ej-chart>

 <code> this.tooltip = 10 | Not Applicable

| Fill color for indicator | **Property:** *fill*

 <ej-chart>
<e-indicators>
<e-indicator [fill]='fill'>
</e-indicator>
</e-indicators>
</ej-chart>

 <code> this.fill = 'red' |

Property: *fill* `

 <ejs-chart>
 <e-indicators>
 <e-indicator [fill]='tooltip' >
 </e-indicator>
 </e-indicators>
 </ejs-chart>

 <code> this.fill = 'blue';`

| Width for indicator | **Property:** *width* `

 <ej-chart>
 <e-indicators>
 <e-indicator [width]='width' >
 </e-indicator>
 </e-indicators>
 </ej-chart>

 <code> this.width = 2` | **Property:** *width* `

 <ejs-chart>
 <e-indicators>
 <e-indicator [width]='width' >
 </e-indicator>
 </e-indicators>
 </ejs-chart>

 <code> this.width = 2;`

| xAxis name for indicator | **Property:** *xAxisName* `

 <ej-chart>
 <e-indicators>
 <e-indicator [xAxisName]='xName' >
 </e-indicator>
 </e-indicators>
 </ej-chart>

 <code> this.xName = 'axis'` | **Property:** *xAxisName* `

 <ejs-chart>
 <e-indicators>
 <e-indicator [xAxisName]='xAxis' >
 </e-indicator>
 </e-indicators>
 </ejs-chart>

 <code> this.xAxis = 'axis';`

| yAxis name for indicator | **Property:** *yAxisName* `

 <ej-chart>
 <e-indicators>
 <e-indicator [yAxisName]='yName' >
 </e-indicator>
 </e-indicators>
 </ej-chart>

 <code> this.yName = 'axis'` | **Property:** *yAxisName* `

 <ejs-chart>
 <e-indicators>
 <e-indicator [yAxisName]='yAxis' >
 </e-indicator>
 </e-indicators>
 </ejs-chart>

 <code> this.yAxis = 'axis';`

| xAxis name for indicator | **Property:** *xAxisName* `

 <ej-chart>
 <e-indicators>
 <e-indicator [xAxisName]='xName' >
 </e-indicator>
 </e-indicators>
 </ej-chart>

 <code> this.xName = 'axis'` | **Property:** *xAxisName* `

 <ejs-chart>
 <e-indicators>
 <e-indicator [xAxisName]='xAxis' >
 </e-indicator>
 </e-indicators>
 </ejs-chart>

 <code> this.xAxis = 'axis';`

| dataSource for indicator | **Property:** *points* `

 <ej-chart>
 <e-indicators>
 <e-indicator [points]='points' >
 </e-indicator>
 </e-indicators>
 </ej-chart>

 <code> this.points = []` | **Property:** *dataSource* `

 <ejs-chart>
 <e-indicators>
 <e-indicator [dataSource]='data' >
 </e-indicator>
 </e-indicators>
 </ejs-chart>

 <code> this.data = 'axis';`

Legend

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Visibility of legend | **Property:** *legend.visible* `

 <ej-chart [legend]='legend' >
 </ej-chart>

 <code> this.legend = { visible: true }` | **Property:** *legend.visible* `

 <ejs-chart [legend]='legend' >
 </ejs-chart>

 <code> this.legend = { visible: true }`

| Height of legend | **Property:** *legend.size.height* `

 <ej-chart [legend]='legend' >
 </ej-chart>

 <code> this.legend = { size: { height: 20 } }` | **Property:** *legend.height* `

 <ejs-chart [legend]='legend' >
 </ejs-chart>

 <code> this.legend = { height: '40' }`

| Width of legend | **Property:** *legend.size.width* `

 <ej-chart [legend]='legend' >
 </ej-chart>

 <code> this.legend = { size: { width: 20 } }` | **Property:** *legend.height* `

 <ejs-chart [legend]='legend' >
 </ejs-chart>

 <code> this.legend = { width: '40' }`

| Location of legend | **Property:** *legend.location*

 <ej-chart [legend]='legend'>
</ej-chart>

 <code> this.legend = { location: { x: 10, y: 30 } } | **Property:** *legend.location*

 <ejs-chart [legend]='legend'>
</ejs-chart>

 <code> this.legend = { location: { x: 10, y: 30 } }

| Padding of legend | **Property:** *legend.padding*

 <ej-chart [legend]='legend'>
</ej-chart>

 <code> this.legend = { padding: 20 } | **Property:** *legend.padding*

 <ejs-chart [legend]='legend'>
</ejs-chart>

 <code> this.legend = { padding: 8 }

| Alignment of legend | **Property:** *legend.alignment*

 <ej-chart [legend]='legend'>
</ej-chart>

 <code> this.legend = { alignment: 'near' } | **Property:** *legend.alignment*

 <ejs-chart [legend]='legend'>
</ejs-chart>

 <code> this.legend = { alignment: 'near' }

| Text style of legend | **Property:** *legend.font*

 <ej-chart [legend]='legend'>
</ej-chart>

 <code> this.legend = { font: { size: '12px', color: 'red' } } | **Property:** *legend.textStyle*

 <ejs-chart [legend]='legend'>
</ejs-chart>

 <code> this.legend = { textStyle: { size: '12px', color: 'red' } }

| Shape height of legend | **Property:** *legend.itemStyle.height*

 <ej-chart [legend]='legend'>
</ej-chart>

 <code> this.legend = { itemStyle: { height: 20 } } | **Property:** *legend.shapeHeight*

 <ejs-chart [legend]='legend'>
</ejs-chart>

 <code> this.legend = { shapeHeight: 20 }

| Shape width of legend | **Property:** *legend.itemStyle.width*

 <ej-chart [legend]='legend'>
</ej-chart>

 <code> this.legend = { itemStyle: { width: 20 } } | **Property:** *legend.shapeWidth*

 <ejs-chart [legend]='legend'>
</ejs-chart>

 <code> this.legend = { shapeWidth: 20 }

| Shape border of legend | **Property:** *legend.itemStyle.border*

 <ej-chart [legend]='legend'>
</ej-chart>

 <code> this.legend = { itemStyle: { border: { width: 2, color: 'red' } } } | **Property:** *legend.shapeBorder*

 <ejs-chart [legend]='legend'>
</ejs-chart>

 <code> this.legend = { shapeBorder: { color: 'red', width: 2 } }

| Shape padding of legend | **Property:** *legend.itemPadding*

 <ej-chart [legend]='legend'>
</ej-chart>

 <code> this.legend = { itemPadding: 10 } } | **Property:** *legend.shapePadding*

 <ejs-chart [legend]='legend'>
</ejs-chart>

 <code> this.legend = { shapePadding: 10 }

| Background of legend | **Property:** *legend.background*

 <ej-chart [legend]='legend'>
</ej-chart>

 <code> this.legend = { background: 'transparent' } } | **Property:** *legend.background*

 <ejs-chart [legend]='legend'>
</ejs-chart>

 <code> this.legend = { background: 'transparent' }

| Opacity of legend | **Property:** *legend.opacity*

 <ej-chart [legend]='legend'>
</ej-chart>

 <code> this.legend = { opacity: 0.7 } } | **Property:** *legend.opacity*

 <ejs-chart [legend]='legend'>
</ejs-chart>

 <code> this.legend = { opacity: 0.6 }

| Toggle visibility of series legend | **Property:** *legend.toggleSeriesVisibility*

 <ej-chart [legend]='legend'>
</ej-chart>

 <code> this.legend = { toggleSeriesVisibility: true } }

Property: *legend.toggleVisibility*

 <ej-chart [legend]='legend'>
</ej-chart>

 <code> this.legend = { toggleVisibility: true }

| Title of legend | **Property:** *legend.title*

 <ej-chart [legend]='legend'>
</ej-chart>

 <code> this.legend = { title: 'legend title' } | **Property:** Not Applicable

| Text over flow of legend | **Property:** *legend.textOverFlow*

 <ej-chart
 [legend]='legend'>
</ej-chart>

 <code> this.legend = { textOverFlow: 'Trim' } |
 Property: *legend.textOverFlow*

 <ej-chart [legend]='legend'>
</ej-chart>

 <code> this.legend = { textStyle:{ textOverFlow: 'Trim' } }

| Maximum text width above to trim | **Property:** *legend.textWidth*

 <ej-chart
 [legend]='legend'>
</ej-chart>

 <code> this.legend = { textWidth: 50 } | **Property:** Not
 Applicable

| Scroll bar for legend | **Property:** *legend.enableScrollBar*

 <ej-chart
 [legend]='legend'>
</ej-chart>

 <code> this.legend = { enableScrollBar: true } |
 Property: Not Applicable

| Row count for legend | **Property:** *legend.rowCount*

 <ej-chart
 [legend]='legend'>
</ej-chart>

 <code> this.legend = { rowCount: 2 } | **Property:** Not
 Applicable

| Column count for legend | **Property:** *legend.columnCount*

 <ej-chart
 [legend]='legend'>
</ej-chart>

 <code> this.legend = { columnCount: 2 } | **Property:**
 Not Applicable

| Fill color for legend | **Property:** *legend.fill*

 <ej-chart [legend]='legend'>
</ej-chart>

 <code> this.legend = { fill: 2 } | **Property:** Not Applicable

PrimaryXAxis

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Default | **Property:** *primaryXAxis*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-
 chart>

 <code> this.primaryXAxis = { } | **Property:** *primaryXAxis*

 <ej-chart
 [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { }

| Alternate grid band for axis | **Property:** *primaryXAxis.alternateGridBand*

 <ej-chart
 [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = {
 alternateGridBand: { even: { fill: 'black' } } } | **Property:** Not Applicable

| Axis line crosses value | **Property:** *primaryXAxis.crossesAt*

 <ej-chart
 [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { crossesAt: 3 }
 | **Property:** *primaryXAxis.crossesAt*

 <ej-chart
 [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { crossesAt: 3
 }

| Axis name in which axis line cross | **Property:** *primaryXAxis.crossInAxis*

 <ej-chart
 [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { crossInAxis:
 'axis' } | **Property:** *primaryXAxis.crossInAxis*

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { crossInAxis: 'axis' } </code>

| Axis elements placing with axis line| **Property:** *primaryXAxis.showNextToAxisLine*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { showNextToAxisLine: true } | **Property:** *primaryXAxis.placeNextToAxisLine*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { placeNextToAxisLine: true } </code>

| Axis line color| **Property:** *primaryXAxis.axisLine.color*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { axisLine: { color: 'red' } } | **Property:** *primaryXAxis.lineStyle.color*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { lineStyle: { color: 'red' } } </code>

| Axis line dashArray| **Property:** *primaryXAxis.axisLine.dashArray*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { axisLine: { dashArray: '10, 5' } } | **Property:** *primaryXAxis.lineStyle.dashArray*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { lineStyle: { dashArray: '10, 5' } } </code>

| Visibility of primaryXAxis| **Property:** *primaryXAxis.axisLine.visible*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { axisLine: { visible: true } } | **Property:** *primaryXAxis.visible*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { visible: true } </code>

| Column index of primaryXAxis| **Property:** *primaryXAxis.columnIndex*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { columnIndex: 2 } | **Property:** *primaryXAxis.columnIndex*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { columnIndex: 2 } </code>

| span of primaryXAxis| **Property:** *primaryXAxis.columnSpan*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { columnSpan: 2 } | **Property:** *primaryXAxis.span*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { span: 2 } </code>

| crosshair visibility of primaryXAxis| **Property:** *primaryXAxis.crosshairLabel*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { crosshairLabel : { visible: true } } | **Property:** *primaryXAxis.crosshairTooltip*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { crosshairTooltip: { enable: true } } </code>

| crosshair fill of primaryXAxis| **Property:** Not Applicable | **Property:** *primaryXAxis.crosshairTooltip*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { crosshairTooltip: { fill: 'blue' } } </code>

|crosshair label text style of primaryXAxis| **Property:** Not Applicable | **Property:**
`primaryXAxis.crosshairTooltip`

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { crosshairTooltip: { labelStyle: { } } }

|Desired interval of primaryXAxis| **Property:** `primaryXAxis.desiredIntervals`

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = {
desiredIntervals: 4 } | **Property:** `primaryXAxis.desiredIntervals`

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = {
desiredIntervals: 4 }

|Edge label placements of primaryXAxis| **Property:** `primaryXAxis.edgeLabelPlacement`

 <ej-
chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = {
edgeLabelPlacement: 'Shift' } | **Property:** `primaryXAxis.edgeLabelPlacement`

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = {
edgeLabelPlacement: 'Hide' }

|Enabling trim of primaryXAxis| **Property:** `primaryXAxis.enableTrim`

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { enableTrim:
true } | **Property:** `primaryXAxis.enableTrim`

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { enableTrim:
true }

|Enabling auto interval while zooming| **Property:** `primaryXAxis.enableAutoIntervalOnZooming`

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis
= { enableAutoIntervalOnZooming: true } | **Property:** `primaryXAxis.enableAutoIntervalOnZooming`

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code>
this.primaryXAxis = { enableAutoIntervalOnZooming: true }

|Font style of primaryXAxis| **Property:** `primaryXAxis.font`

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { font: { } } |
Property: `primaryXAxis.labelStyle`

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-
chart>

 <code> this.primaryXAxis = { labelStyle:{ } }

|Enabling indexed of primaryXAxis| **Property:** `primaryXAxis.indexed`

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { indexed:
true } | **Property:** `primaryXAxis.indexed`

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { indexed:
true }

|IntervalType of primaryXAxis| **Property:** `primaryXAxis.intervalType`

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { intervalType:
'Auto' } | **Property:** `primaryXAxis.intervalType`

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = {
intervalType: 'Auto' }

|Enabling inversed of primaryXAxis| **Property:** `primaryXAxis.isInversed`

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { isInversed:
true } | **Property:** `primaryXAxis.isInversed`

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { isInversed: true }

| Custom label format of primaryXAxis| **Property:** *primaryXAxis.isInversed*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { labelFormat: '{value}K' } | **Property:** *primaryXAxis.labelFormat*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {labelFormat: '{value}K' }

| Label intersect action of primaryXAxis| **Property:** *primaryXAxis.labelIntersectAction*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { labelIntersectAction: 'trim' } | **Property:** *primaryXAxis.labelIntersectAction*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {labelIntersectAction: 'Trim' }

| Label position of primaryXAxis| **Property:** *primaryXAxis.labelPosition*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { labelPosition: 'Inside' } | **Property:** *primaryXAxis.labelPosition*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {labelPosition: 'Inside' }

| Label placement of primaryXAxis| **Property:** *primaryXAxis.labelPlacement*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { labelPlacements: 'OnTicks' } | **Property:** *primaryXAxis.labelPlacement*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {lablePlacement: 'OnTicks' }

| Label alignment of primaryXAxis| **Property:** *primaryXAxis.alignment*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { alignment: 'middle' } | **Property:** Not Applicable

| Label rotation of primaryXAxis| **Property:** *primaryXAxis.labelRotation*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { labelRotation: 45 } | **Property:** *primaryXAxis.labelRotation*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { labelRotation: 45 }

| Log base of primaryXAxis| **Property:** *primaryXAxis.logBase*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { logBase: 4 } | **Property:** *primaryXAxis.logBase*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { logBase: 5 }

| Log base of primaryXAxis| **Property:** *primaryXAxis.logBase*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { logBase: 4 } | **Property:** *primaryXAxis.logBase*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { logBase: 5 }

| Major grid line width of primaryXAxis| **Property:** *primaryXAxis.majorGridLines.width*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { majorGridLines: { width: 2 } } | **Property:** *primaryXAxis.majorGridLines.width*

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {
majorGridLines: { width: 2 } }

| Major grid line color of primaryXAxis | **Property:** *primaryXAxis.majorGridLines.color*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {
majorGridLines: { color: 'red' } } | **Property:** *primaryXAxis.majorGridLines.color*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {
majorGridLines: { color: 'red' } }

| Major grid line dash array of primaryXAxis | **Property:** *primaryXAxis.majorGridLines.dashArray*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {
majorGridLines: { dashArray: '10, 5' } } | **Property:** *primaryXAxis.majorGridLines.dashArray*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code>
this.primaryXAxis = { majorGridLines: { dashArray: '10, 5' } }

| Major grid line opacity of primaryXAxis | **Property:** *primaryXAxis.majorGridLines.opacity*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {
majorGridLines: { opacity: 0.7 } } | **Property:** Not Applicable

| Major grid line visibility of primaryXAxis | **Property:** *primaryXAxis.majorGridLines.visible*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {
majorGridLines: { visible: true } } | **Property:** Not Applicable

| minor grid line width of primaryXAxis | **Property:** *primaryXAxis.minorGridLines.width*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {
minorGridLines: { width: 2 } } | **Property:** *primaryXAxis.minorGridLines.width*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {
minorGridLines: { width: 2 } }

| minor grid line color of primaryXAxis | **Property:** *primaryXAxis.minorGridLines.color*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {
minorGridLines: { color: 'red' } } | **Property:** *primaryXAxis.minorGridLines.color*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {
minorGridLines: { color: 'red' } }

| minor grid line dash array of primaryXAxis | **Property:** *primaryXAxis.minorGridLines.dashArray*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {
minorGridLines: { dashArray: '10, 5' } } | **Property:** *primaryXAxis.minorGridLines.dashArray*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code>
this.primaryXAxis = { minorGridLines: { dashArray: '10, 5' } }

| minor grid line opacity of primaryXAxis | **Property:** *primaryXAxis.minorGridLines.opacity*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {
minorGridLines: { opacity: 0.7 } } | **Property:** Not Applicable

| minor grid line visibility of primaryXAxis | **Property:** *primaryXAxis.minorGridLines.visible*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {
minorGridLines: { visible: true } } | **Property:** Not Applicable

| major Tick line width of primaryXAxis | **Property:** *primaryXAxis.majorTickLines.width*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {

majorTickLines: { width: 2 } } | **Property:** *primaryXAxis.majorTickLines.width*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { majorTickLines: { width: 2 } }

| major Tick line color of primaryXAxis | **Property:** *primaryXAxis.majorTickLines.color*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { majorTickLines: { color: 'red' } } } | **Property:** *primaryXAxis.majorTickLines.color*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { majorTickLines: { color: 'red' } }

| major Tick line opacity of primaryXAxis | **Property:** *primaryXAxis.majorTickLines.opacity*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { majorTickLines: { opacity: 0.7 } } } | **Property:** Not Applicable

| major Tick line visibility of primaryXAxis | **Property:** *primaryXAxis.majorTickLines.visible*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { majorTickLines: { visible: true } } } | **Property:** Not Applicable

| minor Tick line width of primaryXAxis | **Property:** *primaryXAxis.minorTickLines.width*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { minorTickLines: { width: 2 } } } | **Property:** *primaryXAxis.minorTickLines.width*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { minorTickLines: { width: 2 } }

| minor Tick line color of primaryXAxis | **Property:** *primaryXAxis.minorTickLines.color*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { minorTickLines: { color: 'red' } } } | **Property:** *primaryXAxis.minorTickLines.color*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { minorTickLines: { color: 'red' } }

| minor Tick line opacity of primaryXAxis | **Property:** *primaryXAxis.minorTickLines.opacity*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { minorTickLines: { opacity: 0.7 } } } | **Property:** Not Applicable

| minor Tick line visibility of primaryXAxis | **Property:** *primaryXAxis.minorTickLines.visible*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { minorTickLines: { visible: true } } } | **Property:** Not Applicable

| Maximum labels of primaryXAxis | **Property:** *primaryXAxis.maximumLabels*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { maximumLabels: 4 } } | **Property:** *primaryXAxis.maximumLabels*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { maximumLabels: 4 }

| Maximum label width of primaryXAxis | **Property:** *primaryXAxis.maximumLabelWidth*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { maximumLabels: 15 } } | **Property:** *primaryXAxis.maximumLabelWidth*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { maximumLabelWidth:15 }

| Minor ticks per interval of primaryXAxis| **Property:** *primaryXAxis.minorTicksPerInterval*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { minorTicksPerInterval: 5 } } | **Property:** *primaryXAxis.minorTicksPerInterval*

 <ejs-chart [primaryXAxis]='primaryXAxis'>
</ejs-chart>

 <code> this.primaryXAxis = { minorTicksPerInterval:5 }

| Name of primaryXAxis| **Property:** *primaryXAxis.name*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { name: 'axis' } } | **Property:** *primaryXAxis.name*

 <ejs-chart [primaryXAxis]='primaryXAxis'>
</ejs-chart>

 <code> this.primaryXAxis = { name:'axis' }

| Plot offset of primaryXAxis| **Property:** *primaryXAxis.plotOffset*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { plotOffset: 10 } } | **Property:** *primaryXAxis.name*

 <ejs-chart [primaryXAxis]='primaryXAxis'>
</ejs-chart>

 <code> this.primaryXAxis = { plotOffset:10 }

| Orientation of primaryXAxis| **Property:** *primaryXAxis.plotOffset*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { orientation: 'Horizontal' } } | **Property:** *primaryXAxis.name*

 <ejs-chart [primaryXAxis]='primaryXAxis'>
</ejs-chart>

 <code> this.primaryXAxis = { orientation: 'horizontal' }

| Minimum of primaryXAxis| **Property:** *primaryXAxis.range.minimum*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { range: { minimum: 10 } } } | **Property:** *primaryXAxis.minimum*

 <ejs-chart [primaryXAxis]='primaryXAxis'>
</ejs-chart>

 <code> this.primaryXAxis = { minimum: 10 }

| maximum of primaryXAxis| **Property:** *primaryXAxis.range.maximum*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { range: { maximum: 10 } } } | **Property:** *primaryXAxis.maximum*

 <ejs-chart [primaryXAxis]='primaryXAxis'>
</ejs-chart>

 <code> this.primaryXAxis = { maximum: 1000 }

| interval of primaryXAxis| **Property:** *primaryXAxis.range.interval*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { range: { interval: 10 } } } | **Property:** *primaryXAxis.interval*

 <ejs-chart [primaryXAxis]='primaryXAxis'>
</ejs-chart>

 <code> this.primaryXAxis = { interval: 10 }

| Range padding of primaryXAxis| **Property:** *primaryXAxis.rangePadding*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { rangePadding: 'Round' } } | **Property:** *primaryXAxis.rangePadding*

 <ejs-chart [primaryXAxis]='primaryXAxis'>
</ejs-chart>

 <code> this.primaryXAxis = { rangePadding: 'Normal' }

| Rounding places of primaryXAxis| **Property:** *primaryXAxis.roundingPlaces*

 <ej-chart [primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { roundingPlaces: 3 } } | **Property:** *primaryXAxis.labelFormat*

 <ejs-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { labelFormat: 'n3' }

| Tick position of primaryXAxis | **Property:** *primaryXAxis.tickPosition*

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { tickPosition: 'Inside' } | **Property:** *primaryXAxis.tickPosition*

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { tickPosition: 'Inside' }

| Scrollbar settings of primaryXAxis | **Property:** *primaryXAxis.scrollBarSettings*

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { scrollBarSettings: { } } | **Property:** Not Applicable.

| Value type of primaryXAxis | **Property:** *primaryXAxis.valueType*

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { valueType: 'Category' } | **Property:** *primaryXAxis.valueType*

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { valueType: 'Category' }

| Visible of primaryXAxis | **Property:** *primaryXAxis.visible*

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { visible: true }

| **Property:** *primaryXAxis.visible*

 <ej-chart [primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { visible: true }

| Zoom factor of primaryXAxis | **Property:** *primaryXAxis.zoomFactor*

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { zoomFactor: 0.7 } | **Property:** *primaryXAxis.zoomFactor*

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { zoomFactor: 0.7 }

| Zoom position of primaryXAxis | **Property:** *primaryXAxis.zoomPosition*

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = {

zoomPosition: 0.7 } | **Property:** *primaryXAxis.zoomPosition*

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { zoomPosition: 0.7 }

| Label border of primaryXAxis | **Property:** *primaryXAxis.labelBorder*

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { labelBorder: { width: 2, color: 'red' } } | **Property:** *primaryXAxis.labelBorder*

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { labelBorder: { color: 'red', width: 2 } }

| Title of primaryXAxis | **Property:** *primaryXAxis.title*

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { title: 'axis title' } | **Property:** *primaryXAxis.title*

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { title: 'axis title' }

| skeleton of primaryXAxis | **Property:** *primaryXAxis.skeleton*

 <ej-chart

[primaryXAxis]='primaryXAxis'></ej-chart>

 <code> this.primaryXAxis = { skeleton:

'yMd' } | **Property:** *primaryXAxis.skeleton*

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { skeleton:
'yMd' }

|skeletonType of primaryXAxis| **Property:** *primaryXAxis.skeletonType*

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = {
skeletonType: 'Date' } | **Property:** *primaryXAxis.skeletonType*

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = {
skeletonType: 'Date' }

|Stripline of primaryXAxis| **Property:** *primaryXAxis.stripLines*

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { stripLines: [{
}] } | **Property:** *primaryXAxis.skeletonType*

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = { stripLines: [
{ }] }

|Multilevel labels of primaryXAxis| **Property:** *primaryXAxis.multiLevelLabels*

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = {
multiLevelLabels: [{ }] } | **Property:** *primaryXAxis.multiLevelLabels*

 <ej-chart
[primaryXAxis]='primaryXAxis'>
</ej-chart>

 <code> this.primaryXAxis = {
multiLevelLabels: [{ }] }

PrimaryYAxis

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Default | **Property:** *primaryYAxis*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-
chart>

 <code> this.primaryYAxis = { } | **Property:** *primaryYAxis*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { }

| Alternate grid band for axis | **Property:** *primaryYAxis.alternateGridBand*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
alternateGridBand: { even: { fill: 'black' } } } | **Property:** Not Applicable

| Axis line crosses value | **Property:** *primaryYAxis.crossesAt*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { crossesAt: 3 }
| **Property:** *primaryYAxis.crossesAt*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { crossesAt: 3
}

| Axis name in which axis line cross | **Property:** *primaryYAxis.crossInAxis*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { crossInAxis:
'axis' } | **Property:** *primaryYAxis.crossInAxis*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { crossInAxis:
'axis' }

| Axis elements placing with axis line | **Property:** *primaryYAxis.showNextToAxisLine*

 <ej-
chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
showNextToAxisLine: true } | **Property:** *primaryYAxis.placeNextToAxisLine*

 <ej-chart

[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
placeNextToAxisLine: true }

| Axis line color| **Property:** *primaryYAxis.axisLine.color*
</ej-chart>

[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { axisLine: {
color: 'red' } } | **Property:** *primaryYAxis.lineStyle.color*
</ej-chart>

[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { lineStyle: {
color: 'red' } } }

| Axis line dashArray| **Property:** *primaryYAxis.axisLine.dashArray*
</ej-chart>

[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { axisLine: {
dashArray: '10, 5' } } | **Property:** *primaryYAxis.lineStyle.dashArray*
</ej-chart>

[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { lineStyle: {
dashArray: '10, 5' } } }

| Visibility of primaryYAxis| **Property:** *primaryYAxis.axisLine.visible*
</ej-chart>

[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { axisLine: {
visible: true } } | **Property:** *primaryYAxis.visible*
</ej-chart>

[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { visible: true
}

| Column index of primaryYAxis| **Property:** *primaryYAxis.columnIndex*
</ej-chart>

[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { columnIndex:
2 } | **Property:** *primaryYAxis.columnIndex*
</ej-chart>

[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
columnIndex: 2 }

| span of primaryYAxis| **Property:** *primaryYAxis.columnSpan*
</ej-chart>

[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { columnSpan:
2 } | **Property:** *primaryYAxis.span*
</ej-chart> [primaryYAxis]='primaryYAxis'>
</ej-
chart>

 <code> this.primaryYAxis = { span: 2 }

| crosshair visibility of primaryYAxis| **Property:** *primaryYAxis.crosshairLabel*
</ej-chart>

[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
crosshairLabel : { visible: true } } | **Property:** *primaryYAxis.crosshairTooltip*
</ej-chart>

[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
crosshairTooltip: { enable: true } }

| crosshair fill of primaryYAxis| **Property:** Not Applicable | **Property:** *primaryYAxis.crosshairTooltip*

</ej-chart> [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code>
this.primaryYAxis = { crosshairTooltip: { fill: 'blue' } }

| crosshair label text style of primaryYAxis| **Property:** Not Applicable | **Property:**

primaryYAxis.crosshairTooltip
</ej-chart> [primaryYAxis]='primaryYAxis'>
</ej-
chart>

 <code> this.primaryYAxis = { crosshairTooltip: { labelStyle: { } } }

| Desired interval of primaryYAxis| **Property:** *primaryYAxis.desiredIntervals*
</ej-chart>

[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
desiredIntervals: 4 } | **Property:** *primaryYAxis.desiredIntervals*
</ej-chart>

[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
desiredIntervals: 4 }</code>

| Edge label placements of primaryYAxis| **Property:** *primaryYAxis.edgeLabelPlacement*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
edgeLabelPlacement: 'Shift' } | **Property:** *primaryYAxis.edgeLabelPlacement*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
edgeLabelPlacement: 'Hide' }</code>

| Enabling trim of primaryYAxis| **Property:** *primaryYAxis.enableTrim*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { enableTrim:
true } | **Property:** *primaryYAxis.enableTrim*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { enableTrim:
true }</code>

| Enabling auto interval while zooming| **Property:** *primaryYAxis.enableAutoIntervalOnZooming*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis
= { enableAutoIntervalOnZooming: true } | **Property:** *primaryYAxis.enableAutoIntervalOnZooming*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code>
this.primaryYAxis = { enableAutoIntervalOnZooming: true }</code>

| Font style of primaryYAxis| **Property:** *primaryYAxis.font*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { font: { } } |
Property: *primaryYAxis.labelStyle*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-
chart>

 <code> this.primaryYAxis = { labelStyle:{ } }</code>

| Enabling indexed of primaryYAxis| **Property:** *primaryYAxis.indexed*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { indexed:
true } | **Property:** *primaryYAxis.indexed*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { indexed:
true }</code>

| IntervalType of primaryYAxis| **Property:** *primaryYAxis.intervalType*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { intervalType:
'Auto' } | **Property:** *primaryYAxis.intervalType*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
intervalType: 'Auto' }</code>

| Enabling inversed of primaryYAxis| **Property:** *primaryYAxis.isInversed*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { isInversed:
true } | **Property:** *primaryYAxis.isInversed*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { isInversed:
true }</code>

| Custom label format of primaryYAxis| **Property:** *primaryYAxis.isInversed*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { labelFormat:
'{value}K' } | **Property:** *primaryYAxis.labelFormat*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {labelFormat:
'{value}K' }</code>

|Label intersect action of primaryYAxis| **Property:** *primaryYAxis.labelIntersectAction*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { labelIntersectAction: 'trim' } | **Property:** *primaryYAxis.labelIntersectAction*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = {labelIntersectAction: 'Trim' }

|Label position of primaryYAxis| **Property:** *primaryYAxis.labelPosition*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { labelPosition: 'Inside' } | **Property:** *primaryYAxis.labelPosition*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = {labelPosition: 'Inside' }

|Label placement of primaryYAxis| **Property:** *primaryYAxis.labelPlacement*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { labelPlacements: 'OnTicks' } | **Property:** *primaryYAxis.labelPlacement*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = {lablePlacement: 'OnTicks' }

|Label alignment of primaryYAxis| **Property:** *primaryYAxis.alignment*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { alignment: 'middle' } | **Property:** Not Applicable

|Label rotation of primaryYAxis| **Property:** *primaryYAxis.labelRotation*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { labelRotation: 45 } | **Property:** *primaryYAxis.labelRotation*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = { labelRotation: 45 }

|Log base of primaryYAxis| **Property:** *primaryYAxis.logBase*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { logBase: 4 } | **Property:** *primaryYAxis.logBase*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = { logBase: 5 }

|Log base of primaryYAxis| **Property:** *primaryYAxis.logBase*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { logBase: 4 } | **Property:** *primaryYAxis.logBase*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = { logBase: 5 }

|Major grid line width of primaryYAxis| **Property:** *primaryYAxis.majorGridLines.width*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { majorGridLines: { width: 2 } } | **Property:** *primaryYAxis.majorGridLines.width*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = { majorGridLines: { width: 2 } }

|Major grid line color of primaryYAxis| **Property:** *primaryYAxis.majorGridLines.color*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { majorGridLines: { color: 'red' } } | **Property:** *primaryYAxis.majorGridLines.color*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = { majorGridLines: { color: 'red' } }

|Major grid line dash array of primaryYAxis| **Property:** *primaryYAxis.majorGridLines.dashArray*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis
 = { majorGridLines: { dashArray: '10, 5' } } | **Property:** *primaryYAxis.majorGridLines.dashArray*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code>
 this.primaryYAxis = { majorGridLines: { dashArray: '10, 5' } }

|Major grid line opacity of primaryYAxis| **Property:** *primaryYAxis.majorGridLines.opacity*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 majorGridLines: { opacity: 0.7 } } | **Property:** Not Applicable

|Major grid line visibility of primaryYAxis| **Property:** *primaryYAxis.majorGridLines.visible*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 majorGridLines: { visible: true } } | **Property:** Not Applicable

|minor grid line width of primaryYAxis| **Property:** *primaryYAxis.minorGridLines.width*

 <ej-
 chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 minorGridLines: { width: 2 } } | **Property:** *primaryYAxis.minorGridLines.width*

 <ejs-chart
 [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = {
 minorGridLines: { width: 2 } }

|minor grid line color of primaryYAxis| **Property:** *primaryYAxis.minorGridLines.color*

 <ej-
 chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 minorGridLines: { color: 'red' } } | **Property:** *primaryYAxis.minorGridLines.color*

 <ejs-chart
 [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = {
 minorGridLines: { color: 'red' } }

|minor grid line dash array of primaryYAxis| **Property:** *primaryYAxis.minorGridLines.dashArray*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis
 = { minorGridLines: { dashArray: '10, 5' } } | **Property:** *primaryYAxis.minorGridLines.dashArray*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code>
 this.primaryYAxis = { minorGridLines: { dashArray: '10, 5' } }

|minor grid line opacity of primaryYAxis| **Property:** *primaryYAxis.minorGridLines.opacity*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 minorGridLines: { opacity: 0.7 } } | **Property:** Not Applicable

|minor grid line visibility of primaryYAxis| **Property:** *primaryYAxis.minorGridLines.visible*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 minorGridLines: { visible: true } } | **Property:** Not Applicable

|major Tick line width of primaryYAxis| **Property:** *primaryYAxis.majorTickLines.width*

 <ej-
 chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 majorTickLines: { width: 2 } } | **Property:** *primaryYAxis.majorTickLines.width*

 <ejs-chart
 [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = {
 majorTickLines: { width: 2 } }

|major Tick line color of primaryYAxis| **Property:** *primaryYAxis.majorTickLines.color*

 <ej-
 chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 majorTickLines: { color: 'red' } } | **Property:** *primaryYAxis.majorTickLines.color*

 <ejs-chart
 [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = {
 majorTickLines: { color: 'red' } }

| major Tick line opacity of primaryYAxis| **Property:** *primaryYAxis.majorTickLines.opacity*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 majorTickLines: { opacity: 0.7 } } | **Property:** Not Applicable

| major Tick line visibility of primaryYAxis| **Property:** *primaryYAxis.majorTickLines.visible*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 majorTickLines: { visible: true } } | **Property:** Not Applicable

| minor Tick line width of primaryYAxis| **Property:** *primaryYAxis.minorTickLines.width*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 minorTickLines: { width: 2 } } | **Property:** *primaryYAxis.minorTickLines.width*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = {
 minorTickLines: { width: 2 } }

| minor Tick line color of primaryYAxis| **Property:** *primaryYAxis.minorTickLines.color*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 minorTickLines: { color: 'red' } } | **Property:** *primaryYAxis.minorTickLines.color*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = {
 minorTickLines: { color: 'red' } }

| minor Tick line opacity of primaryYAxis| **Property:** *primaryYAxis.minorTickLines.opacity*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 minorTickLines: { opacity: 0.7 } } | **Property:** Not Applicable

| minor Tick line visibility of primaryYAxis| **Property:** *primaryYAxis.minorTickLines.visible*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 minorTickLines: { visible: true } } | **Property:** Not Applicable

| Maximum labels of primaryYAxis| **Property:** *primaryYAxis.maximumLabels*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 maximumLabels: 4 } } | **Property:** *primaryYAxis.maximumLabels*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = {
 maximumLabels: 4 }

| Maximum label width of primaryYAxis| **Property:** *primaryYAxis.maximumLabelWidth*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 maximumLabels: 15 } } | **Property:** *primaryYAxis.maximumLabelWidth*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = {
 maximumLabelWidth:15 }

| Minor ticks per interval of primaryYAxis| **Property:** *primaryYAxis.minorTicksPerInterval*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 minorTicksPerInterval: 5 } } | **Property:** *primaryYAxis.minorTicksPerInterval*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = {
 minorTicksPerInterval:5 }

| Name of primaryYAxis| **Property:** *primaryYAxis.name*

 <ej-chart [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { name: 'axis' } }
 | **Property:** *primaryYAxis.name*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = { name:'axis' }

|Plot offset of primaryYAxis| **Property:** *primaryYAxis.plotOffset*

 <ej-chart
 [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { plotOffset:
 10 } } | **Property:** *primaryYAxis.name*

 <ejs-chart
 [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = {
 plotOffset:10 }

|Orientation of primaryYAxis| **Property:** *primaryYAxis.plotOffset*

 <ej-chart
 [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { orientation:
 'Horizontal' } } | **Property:** *primaryYAxis.name*

 <ejs-chart
 [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = { orientation:
 'horizontal' }

|Minimum of primaryYAxis| **Property:** *primaryYAxis.range.minimum*

 <ej-chart
 [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { range: {
 minimum: 10 } } } | **Property:** *primaryYAxis.minimum*

 <ejs-chart
 [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = { minimum:
 10 }

|maximum of primaryYAxis| **Property:** *primaryYAxis.range.maximum*

 <ej-chart
 [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { range: {
 maximum: 10 } } } | **Property:** *primaryYAxis.maximum*

 <ejs-chart
 [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = { maximum:
 1000 }

|interval of primaryYAxis| **Property:** *primaryYAxis.range.interval*

 <ej-chart
 [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { range: {
 interval: 10 } } } | **Property:** *primaryYAxis.interval*

 <ejs-chart
 [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = { interval: 10 }

|Range padding of primaryYAxis| **Property:** *primaryYAxis.rangePadding*

 <ej-chart
 [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 rangePadding: 'Round' } } | **Property:** *primaryYAxis.rangePadding*

 <ejs-chart
 [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = {
 rangePadding: 'Normal' }

|Rounding places of primaryYAxis| **Property:** *primaryYAxis.roundingPlaces*

 <ej-chart
 [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
 roundingPlaces: 3 } } | **Property:** *primaryYAxis.labelFormat*

 <ejs-chart
 [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = { labelFormat:
 'n3' }

|Tick position of primaryYAxis| **Property:** *primaryYAxis.tickPosition*

 <ej-chart
 [primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { tickPosition:
 'Inside' } } | **Property:** *primaryYAxis.tickPosition*

 <ejs-chart
 [primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = { tickPosition:
 'Inside' }

|Scrollbar settings of primaryYAxis| **Property:** *primaryYAxis.scrollBarSettings*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
scrollBarSettings: { }} | **Property:** Not Applicable.

|Value type of primaryYAxis| **Property:** *primaryYAxis.valueType*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { valueType:
'Category' } | **Property:** *primaryYAxis.valueType*

 <ejs-chart
[primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = { valueType:
'Category' }

|Visible of primaryYAxis| **Property:** *primaryYAxis.visible*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { visible: true }
| **Property:** *primaryYAxis.visible*

 <ejs-chart [primaryYAxis]='primaryYAxis'>
</ejs-
chart>

 <code> this.primaryYAxis = { visible: true }

|Zoom factor of primaryYAxis| **Property:** *primaryYAxis.zoomFactor*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { zoomFactor:
0.7 } | **Property:** *primaryYAxis.zoomFactor*

 <ejs-chart
[primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = { zoomFactor:
0.7 }

|Zoom position of primaryYAxis| **Property:** *primaryYAxis.zoomPosition*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = {
zoomPosition: 0.7 } | **Property:** *primaryYAxis.zoomPosition*

 <ejs-chart
[primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = {
zoomPosition: 0.7 }

|Label border of primaryYAxis| **Property:** *primaryYAxis.labelBorder*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { labelBorder: {
width: 2, color: 'red' } } | **Property:** *primaryYAxis.labelBorder*

 <ejs-chart
[primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = { labelBorder:
{ color: 'red', width: 2 } }

|Title of primaryYAxis| **Property:** *primaryYAxis.title*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { title: ' axis
title' } | **Property:** *primaryYAxis.title*

 <ejs-chart
[primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = { title: 'axis
title' }

|skeleton of primaryYAxis| **Property:** *primaryYAxis.skeleton*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { skeleton:
'yMd' } | **Property:** *primaryYAxis.skeleton*

 <ejs-chart
[primaryYAxis]='primaryYAxis'>
</ejs-chart>

 <code> this.primaryYAxis = { skeleton:
'yMd' }

|skeletonType of primaryYAxis| **Property:** *primaryYAxis.skeletonType*

 <ej-chart
[primaryYAxis]='primaryYAxis'>
</ej-chart>

 <code> this.primaryYAxis = { skeletonType:
'Date' } | **Property:** *primaryYAxis.skeletonType*

 <ejs-chart

[primaryYAxis]='primaryYAxis'></ej-chart>

 <code> this.primaryYAxis = {
skeletonType: 'Date' }

| Stripline of primaryYAxis| **Property:** *primaryYAxis.striplines*

 <ej-chart>

[primaryYAxis]='primaryYAxis'></ej-chart>

 <code> this.primaryYAxis = { striplines: [{
}] } | **Property:** *primaryYAxis.skeletonType*

 <ej-chart>

[primaryYAxis]='primaryYAxis'></ej-chart>

 <code> this.primaryYAxis = { striplines: [{
}] }

| Multilevel labels of primaryYAxis| **Property:** *primaryYAxis.multiLevelLabels*

 <ej-chart>

[primaryYAxis]='primaryYAxis'></ej-chart>

 <code> this.primaryYAxis = {
multiLevelLabels: [{ }] } | **Property:** *primaryYAxis.multiLevelLabels*

 <ej-chart>

[primaryYAxis]='primaryYAxis'></ej-chart>

 <code> this.primaryYAxis = {
multiLevelLabels: [{ }] }

Axes

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Default | **Property:** *axis*

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>

</e-axes>
</ej-chart> | **Property:** *axis*

 <ej-chart>
 <e-axes>
 <e-axis>

 </e-axis>
 </e-axes>
</ej-chart>

| Alternate grid band for axis | **Property:** *axis.alternateGridBand*

 <ej-chart>
 <e-axes>

 <e-axis [alternateGridBand]='band'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.band = { even:{ fill: 'black' } } | **Property:** Not Applicable

| Axis line crosses value | **Property:** *axis.crossesAt*

 <ej-chart>
 <e-axes>
 <e-axis
[crossesAt]='cross'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.cross = 3 }
| **Property:** *axis.crossesAt*

 <ej-chart>
 <e-axes>
 <e-axis [crossesAt]='cross'>

 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.cross = 3

| Axis name in which axis line cross | **Property:** *axis.crossInAxis*

 <ej-chart>
 <e-axes>

 <e-axis crossInAxis='axis'>
 </e-axis>
 </e-axes>
</ej-chart>

 |
Property: *axis.crossInAxis*

 <ej-chart>
 <e-axes>
 <e-axis crossInAxis='axis'>

 </e-axis>
 </e-axes>
</ej-chart>

| Axis elements placing with axis line| **Property:** *axis.showNextToAxisLine*

 <ej-chart>

<e-axes>
 <e-axis [showNextToAxisLine]='value'>
 </e-axis>
 </e-axes>
</ej-
chart>

 <code> this.value = true } | **Property:** *axis.placeNextToAxisLine*

 <ej-
chart>
 <e-axes>
 <e-axis [placeNextToAxisLine]='place'>
 </e-axis>
 </e-axes>

</ej-chart>

 <code>this.place = true |

| Axis line color| **Property:** *axis.axisLine.color*

 <ej-chart>
 <e-axes>
 <e-axis
[axisLine]='line'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.line = { color:
'red' } | **Property:** *axis.lineStyle.color*

 <ej-chart>
 <e-axes>
 <e-axis
[lineStyle]='line'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.line = {
color: 'red' } |

| Axis line dashArray| **Property:** *axis.axisLine.dashArray*

 <ej-chart>
 <e-axes>
 <e-axis [axisLine]='line'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.line = { dashArray: '10, 5' } | **Property:** *axis.lineStyle.dashArray*

 <ejs-chart>
 <e-axes>
 <e-axis [lineStyle]='line'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.line = dashArray: '10, 5' }

| Visibility of axis| **Property:** *axis.axisLine.visible*

 <ej-chart>
 <e-axes>
 <e-axis [axisLine]='line'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.line = { visible: true } } | **Property:** *axis.visible*

 <ejs-chart>
 <e-axes>
 <e-axis [visible]='visible'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.visible: true |

| ColumnIndex of axis| **Property:** *axis.columnIndex*

 <ej-chart>
 <e-axes>
 <e-axis [columnIndex]='column'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.column = 2 | **Property:** *axis.columnIndex*

 <ejs-chart>
 <e-axes>
 <e-axis [columnIndex]='column'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.column = 2 |

| span of axis| **Property:** *axis.columnSpan*

 <ej-chart>
 <e-axes>
 <e-axis [columnSpan]='span'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.span = 2 | **Property:** *axis.span*

 <ejs-chart>
 <e-axes>
 <e-axis [span]='span'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.span = 2 |

| crosshair visibility of axis| **Property:** *axis.crosshairLabel*

 <ej-chart>
 <e-axes>
 <e-axis [crosshairLabel]='label'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.label = { visible: true } | **Property:** *axis.crosshairTooltip*

 <ejs-chart>
 <e-axes>
 <e-axis [crosshairTooltip]='crosshair'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.crosshair = { enable: true }

| crosshair fill of axis| **Property:** Not Applicable | **Property:** *axis.crosshairTooltip.fill*

 <ejs-chart>
 <e-axes>
 <e-axis [crosshairTooltip]='tooltip'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.tooltip = { fill: 'blue' } |

| crosshair label text style of axis| **Property:** Not Applicable | **Property:** *axis.crosshairTooltip*

 <ejs-chart>
 <e-axes>
 <e-axis [crosshairTooltip]='tooltip'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.tooltip = { labelStyle: { } }

| Desired interval of axis| **Property:** *axis.desiredIntervals*

 <ej-chart>
 <e-axes>
 <e-axis [desiredIntervals]='interval'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.interval = 4 | **Property:** *axis.desiredIntervals*

 <ejs-chart>
 <e-axes>
 <e-axis [desiredIntervals]='interval'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.interval = 4 |

| Edge label placements of axis| **Property:** *axis.edgeLabelPlacement*

 <ej-chart>
 <e-axes>
 <e-axis edgeLabelPlacement='Shift'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.edgeLabelPlacement = 'Shift' | **Property:** *axis.edgeLabelPlacement*

 <ejs-chart>
 <e-axes>
 <e-axis edgeLabelPlacement='Shift'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.edgeLabelPlacement = 'Shift' |

| Enabling trim of axis| **Property:** *axis.enableTrim*

 <ej-chart>
 <e-axes>
 <e-axis [enableTrim]='trim'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.trim =

true | **Property:** *axis.enableTrim*

 <ej-chart>
 <e-axes>
 <e-axis
[enableTrim]='trim'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.trim =
true

| Enabling auto interval while zooming| **Property:** *axis.enableAutoIntervalOnZooming*

 <ej-
chart>
 <e-axes>
 <e-axis [enableAutoIntervalOnZooming]='zoom'>
 </e-axis>

</e-axes>
</ej-chart>

 <code> this.zoom = true | **Property:**
axis.enableAutoIntervalOnZooming

 <ej-chart>
 <e-axes>
 <e-axis
[enableAutoIntervalOnZooming]='zoom'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.zoom = true

| Font style of axis| **Property:** *axis.font*

 <ej-chart>
 <e-axes>
 <e-axis
[font]='font'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.font = { } |
Property: *axis.labelStyle*

 <ej-chart>
 <e-axes>
 <e-axis [labelStyle]='label'>

 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.label = { }

| Enabling indexed of axis| **Property:** *axis.indexed*

 <ej-chart>
 <e-axes>
 <e-axis
[indexed]='indexed'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.indexed
= true | **Property:** *axis.indexed*

 <ej-chart>
 <e-axes>
 <e-axis
[indexed]='indexed'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code>
this.indexed = true |

| IntervalType of axis| **Property:** *axis.intervalType*

 <ej-chart>
 <e-axes>
 <e-axis
intervalType='Years'>
 </e-axis>
 </e-axes>
</ej-chart> | **Property:** *axis.intervalType*

 <ej-chart>
 <e-axes>
 <e-axis intervalType='Years'>
 </e-axis>
 </e-
axes>
</ej-chart>

| Enabling inversed of axis| **Property:** *axis.isInversed*

 <ej-chart>
 <e-axes>
 <e-
axis [isInversed]='inverse'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code>
this.inverse = true | **Property:** *axis.isInversed*

 <ej-chart>
 <e-axes>
 <e-axis
[isInversed]='inverse'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code>
this.inverse = true

| Custom label format of axis| **Property:** *axis.isInversed*

 <ej-chart>
 <e-axes>
 <e-
axis labelFormat='{value}K'>
 </e-axis>
 </e-axes>
</ej-chart> | **Property:**
axis.labelFormat

 <ej-chart>
 <e-axes>
 <e-axis labelFormat='{value}K'>

</e-axis>
 </e-axes>
</ej-chart>|

| Label intersect action of axis| **Property:** *axis.labelIntersectAction*

 <ej-chart>
 <e-
axes>
 <e-axis labelIntersectAction='trim'>
 </e-axis>
 </e-axes>
</ej-chart> |
Property: *axis.labelIntersectAction*

 <ej-chart>
 <e-axes>
 <e-axis
labelIntersectAction='Trim'>
 </e-axis>
 </e-axes>
</ej-chart>|

| Label position of axis| **Property:** *axis.labelPosition*

 <ej-chart>
 <e-axes>
 <e-axis
labelPosition='Inside'>
 </e-axis>
 </e-axes>
</ej-chart> | **Property:** *axis.labelPosition*

 <ej-chart>
 <e-axes>
 <e-axis labelPosition='Outside'>
 </e-axis>

</e-axes>
</ej-chart>

|Label placement of axis| **Property:** *axis.labelPlacement*

 <ej-chart>
 <e-axes>
 <e-axis labelPlacement='OnTicks'>
 </e-axis>
 </e-axes>
</ej-chart> | **Property:** *axis.labelPlacement*

 <ejs-chart>
 <e-axes>
 <e-axis labelPlacement='BetweenTicks'>
 </e-axis>
 </e-axes>
</ejs-chart>

|Label alignment of axis| **Property:** *axis.alignment*

 <ej-chart>
 <e-axes>
 <e-axis labelAlignment='middle'>
 </e-axis>
 </e-axes>
</ej-chart> | **Property:** Not Applicable

|Label rotation of axis| **Property:** *axis.labelRotation*

 <ej-chart>
 <e-axes>
 <e-axis [labelRotation]='rotation'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.rotation = 45 | **Property:** *axis.labelRotation*

 <ejs-chart>
 <e-axes>
 <e-axis [labelRotation]='rotation'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.rotation = 45 |

|Log base of axis| **Property:** *axis.logBase*

 <ej-chart>
 <e-axes>
 <e-axis [logBase]='log'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.log = 4 | **Property:** *axis.logBase*

 <ejs-chart>
 <e-axes>
 <e-axis [logBase]='log'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.log = 5

|Major grid line width of axis| **Property:** *axis.majorGridLines.width*

 <ej-chart>
 <e-axes>
 <e-axis [majorGridLines]='majorGridLines'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.majorGridLines = { width: 2 } | **Property:** *axis.majorGridLines.width*

 <ejs-chart>
 <e-axes>
 <e-axis [majorGridLines]='majorGridLines'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.majorGridLines = { width: 2 } |

|Major grid line color of axis| **Property:** *axis.majorGridLines.color*

 <ej-chart>
 <e-axes>
 <e-axis [majorGridLines]='majorGridLines'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.majorGridLines = { color: 'red' } | **Property:** *axis.majorGridLines.color*

 <ejs-chart>
 <e-axes>
 <e-axis [majorGridLines]='majorGridLines'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.majorGridLines = { color: 'red' }

|Major grid line dash array of axis| **Property:** *axis.majorGridLines.dashArray*

 <ej-chart>
 <e-axes>
 <e-axis [majorGridLines]='majorGridLines'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.majorGridLines = { dashArray: '10, 5' } | **Property:** *axis.majorGridLines.dashArray*

 <ejs-chart>
 <e-axes>
 <e-axis [majorGridLines]='majorGridLines'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.majorGridLines = { dashArray: '10, 5' } |

|Major grid line opacity of axis| **Property:** *axis.majorGridLines.opacity*

 <ej-chart>
 <e-axes>
 <e-axis [majorGridLines]='majorGridLines'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.majorGridLines= { opacity: 0.7 } | **Property:** Not Applicable

|Major grid line visibility of axis| **Property:** *axis.majorGridLines.visible*

 <ej-chart>
 <e-axes>
 <e-axis [majorGridLines]='majorGridLines'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.majorGridLines = { visible: true } | **Property:** Not Applicable

|minor grid line width of axis| **Property:** *axis.minorGridLines.width*

 <ej-chart>
 <e-axes>
 <e-axis [minorGridLines]='minorGridLines'>
 </e-axis>
 </e-axes>
</ej-

chart>

 <code> this.minorGridLines = { width: 2 } | **Property:** *axis.minorGridLines.width*

 <ej-chart>
 <e-axes>
 <e-axis [minorGridLines]='minorGridLines'>
 </e-axis>
 </e-axes>
 </ej-chart>

 <code> this.minorGridLines = { width: 2 }

| minor grid line color of axis | **Property:** *axis.minorGridLines.color*

 <ej-chart>
 <e-axes>
 <e-axis [minorGridLines]='minorGridLines'>
 </e-axis>
 </e-axes>
 </ej-chart>

 <code> this.minorGridLines= { color: 'red' } | **Property:** *axis.minorGridLines.color*

 <ej-chart>
 <e-axes>
 <e-axis [minorGridLines]='minorGridLines'>
 </e-axis>
 </e-axes>
 </ej-chart>

 <code> this.minorGridLines = { color: 'red' }

| minor grid line dash array of axis | **Property:** *axis.minorGridLines.dashArray*

 <ej-chart>
 <e-axes>
 <e-axis [minorGridLines]='minorGridLines'>
 </e-axis>
 </e-axes>
 </ej-chart>

 <code> this.minorGridLines = { dashArray: '10, 5' } | **Property:** *axis.minorGridLines.dashArray*

 <ej-chart>
 <e-axes>
 <e-axis [minorGridLines]='minorGridLines'>
 </e-axis>
 </e-axes>
 </ej-chart>

 <code> this.minorGridLines = { dashArray: '10, 5' }

| minor grid line opacity of axis | **Property:** *axis.minorGridLines.opacity*

 <ej-chart>
 <e-axes>
 <e-axis [minorGridLines]='minorGridLines'>
 </e-axis>
 </e-axes>
 </ej-chart>

 <code> this.minorGridLines = { opacity: 0.7 } | **Property:** Not Applicable

| minor grid line visibility of axis | **Property:** *axis.minorGridLines.visible*

 <ej-chart>
 <e-axes>
 <e-axis [minorGridLines]='minorGridLines'>
 </e-axis>
 </e-axes>
 </ej-chart>

 <code> this.minorGridLines = { visible: true } | **Property:** Not Applicable

| major Tick line width of axis | **Property:** *axis.majorTickLines.width*

 <ej-chart>
 <e-axes>
 <e-axis [majorTickLines]='majorTickLines'>
 </e-axis>
 </e-axes>
 </ej-chart>

 <code> this.majorTickLines = { width: 2 } | **Property:** *axis.majorTickLines.width*

 <ej-chart>
 <e-axes>
 <e-axis [majorTickLines]='majorTickLines'>
 </e-axis>
 </e-axes>
 </ej-chart>

 <code> this.majorTickLines: { width: 2 } |

| major Tick line color of axis | **Property:** *axis.majorTickLines.color*

 <ej-chart>
 <e-axes>
 <e-axis [majorTickLines]='majorTickLines'>
 </e-axis>
 </e-axes>
 </ej-chart>

 <code> this.majorTickLines = { color: 'red' } | **Property:** *axis.majorTickLines.color*

 <ej-chart>
 <e-axes>
 <e-axis [majorTickLines]='majorTickLines'>
 </e-axis>
 </e-axes>
 </ej-chart>

 <code> this.majorTickLines = { color: 'red' }

| major Tick line opacity of axis | **Property:** *axis.majorTickLines.opacity*

 <ej-chart>
 <e-axes>
 <e-axis [majorTickLines]='majorTickLines'>
 </e-axis>
 </e-axes>
 </ej-chart>

 <code> this.majorTickLines = { opacity: 0.7 } | **Property:** Not Applicable

| major Tick line visibility of axis | **Property:** *axis.majorTickLines.visible*

 <ej-chart>
 <e-axes>
 <e-axis [majorTickLines]='majorTickLines'>
 </e-axis>
 </e-axes>
 </ej-chart>

 <code> this.majorTickLines = { visible: true } | **Property:** Not Applicable

| minor Tick line width of axis | **Property:** *axis.minorTickLines.width*

 <ej-chart>
 <e-axes>
 <e-axis [minorTickLines]='minorTickLines'>
 </e-axis>
 </e-axes>
 </ej-chart>

 <code> this.minorTickLines = { width: 2 } | **Property:** *axis.minorTickLines.width*


```
</> <ej-chart><br> <e-axes> <br> <e-axis [minorTickLines]='minorTickLines'> <br> </e-axis>
<br> </e-axes> <br></ej-chart> <br> <br> <code> this.minorTickLines = { width: 2 } }
```

| minor Tick line color of axis | **Property:** *axis.minorTickLines.color*

 <ej-chart>
 <e-axes>
 <e-axis [minorTickLines]='minorTickLines'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.minorTickLines = { color: 'red' } | **Property:** *axis.minorTickLines.color*

 <ej-chart>
 <e-axes>
 <e-axis [minorTickLines]='minorTickLines'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.minorTickLines = { color: 'red' }

| minor Tick line opacity of axis | **Property:** *axis.minorTickLines.opacity*

 <ej-chart>
 <e-axes>
 <e-axis [minorTickLines]='minorTickLines'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.minorTickLines = { opacity: 0.7 } | **Property:** Not Applicable

| minor Tick line visibility of axis | **Property:** *axis.minorTickLines.visible*

 <ej-chart>
 <e-axes>
 <e-axis [minorTickLines]='minorTickLines'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.minorTickLines = { visible: true } | **Property:** Not Applicable

| Maximum labels of axis | **Property:** *axis.maximumLabels*

 <ej-chart>
 <e-axes>
 <e-axis [maximumLabels]='labels'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.labels = 4 | **Property:** *axis.maximumLabels*

 <ej-chart>
 <e-axes>
 <e-axis [maximumLabels]='labels'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.labels = 4

| Maximum label width of axis | **Property:** *axis.maximumLabelWidth*

 <ej-chart>
 <e-axes>
 <e-axis [maximumLabelWidth]='width'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.width = 15 | **Property:** *axis.maximumLabelWidth*

 <ej-chart>
 <e-axes>
 <e-axis [maximumLabelWidth]='width'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.width = 15;

| Minor ticks per interval of axis | **Property:** *axis.minorTicksPerInterval*

 <ej-chart>
 <e-axes>
 <e-axis [minorTicksPerInterval]='interval'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.interval = 5 | **Property:** *axis.minorTicksPerInterval*

 <ej-chart>
 <e-axes>
 <e-axis [minorTicksPerInterval]='interval'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.interval= 5 |

| Name of axis | **Property:** *axis.name*

 <ej-chart>
 <e-axes>
 <e-axis name='XAxisName'>
 </e-axis>
 </e-axes>
</ej-chart> | **Property:** *axis.name*

 <ej-chart>
 <e-axes>
 <e-axis name='xAxis'>
 </e-axis>
 </e-axes>
</ej-chart> |

| Plot offset of axis | **Property:** *axis.plotOffset*

 <ej-chart>
 <e-axes>
 <e-axis [plotOffset]='offset'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.offset = 10 | **Property:** *axis.plotOffset*

 <ej-chart>
 <e-axes>
 <e-axis [plotOffset]='offset'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.axis = { plotOffset:10 }

| Orientation of axis | **Property:** *axis.orientation*

 <ej-chart>
 <e-axes>
 <e-axis orientation='horizontal'>
 </e-axis>
 </e-axes>
</ej-chart> | **Property:**

axis.orientation

 <ejs-chart>
 <e-axes>
 <e-axis orientation='horizontal'>
 </e-axis>
 </e-axes>
</ejs-chart> |

| Minimum of axis| **Property:** *axis.range.minimum*

 <ej-chart>
 <e-axes>
 <e-axis [range]='range'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.range: { minimum: 10 } | **Property:** *axis.minimum*

 <ejs-chart>
 <e-axes>
 <e-axis minimum='minimum'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.minimum = 10

| maximum of axis| **Property:** *axis.range.maximum*

 <ej-chart>
 <e-axes>
 <e-axis [range]='range'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this. range: { maximum: 10 } | **Property:** *axis.maximum*

 <ejs-chart>
 <e-axes>
 <e-axis [maximum]='maximum'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.maximum = 1000 |

| interval of axis| **Property:** *axis.range.interval*

 <ej-chart>
 <e-axes>
 <e-axis [range]='range'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.range = { interval: 10 } | **Property:** *axis.interval*

 <ejs-chart>
 <e-axes>
 <e-axis interval='interval'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.interval: 10 |

| Range padding of axis| **Property:** *axis.rangePadding*

 <ej-chart>
 <e-axes>
 <e-axis rangePadding='Additional'>
 </e-axis>
 </e-axes>
</ej-chart> | **Property:** *axis.rangePadding*

 <ejs-chart>
 <e-axes>
 <e-axis rangePadding='Round'>
 </e-axis>
 </e-axes>
</ejs-chart>

| Rounding places of axis| **Property:** *axis.roundingPlaces*

 <ej-chart>
 <e-axes>
 <e-axis [roundingPlaces]='round'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.round = 3 } | **Property:** *axis.labelFormat*

 <ejs-chart>
 <e-axes>
 <e-axis labelFormat='n3'>
 </e-axis>
 </e-axes>
</ejs-chart>

| Tick position of axis| **Property:** *axis.tickPosition*

 <ej-chart>
 <e-axes>
 <e-axis tickPosition='Inside'>
 </e-axis>
 </e-axes>
</ej-chart> | **Property:** *axis.tickPosition*

 <ejs-chart>
 <e-axes>
 <e-axis tickPosition='Inside'>
 </e-axis>
 </e-axes>
</ejs-chart>

| Scrollbar settings of axis| **Property:** *axis.scrollBarSettings*

 <ej-chart>
 <e-axes>
 <e-axis [scrollBarSettings]='scroll'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.scroll = { } | **Property:** Not Applicable.

| Value type of axis| **Property:** *axis.valueType*

 <ej-chart>
 <e-axes>
 <e-axis valueType='Category'>
 </e-axis>
 </e-axes>
</ej-chart> | **Property:** *axis.valueType*

 <ejs-chart>
 <e-axes>
 <e-axis valueType='Logarithmic'>
 </e-axis>
 </e-axes>
</ejs-chart>

| Visible of axis| **Property:** *axis.visible*

 <ej-chart>
 <e-axes>
 <e-axis [visible]='visible'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.visible: true | **Property:** *axis.visible*

 <ejs-chart>
 <e-axes>
 <e-axis [visible]='visible'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.visible: true

| Zoom factor of axis | **Property:** *axis.zoomFactor*

 <ej-chart>
 <e-axes>
 <e-axis
 [zoomFactor]='zoom'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code> this.zoom =
 0.7 | **Property:** *axis.zoomFactor*

 <ejs-chart>
 <e-axes>
 <e-axis
 [zoomFactor]='zoom'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.zoom
 = 0.7 |

| Zoom position of axis | **Property:** *axis.zoomPosition*

 <ej-chart>
 <e-axes>
 <e-
 axis [zoomPosition]='zoom'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code>
 this.zoom = 0.7 | **Property:** *axis.zoomPosition*

 <ejs-chart>
 <e-axes>
 <e-axis
 [zoomPosition]='zoom'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code>
 this.zoom = 0.7 |

| Label border of axis | **Property:** *axis.labelBorder*

 <ej-chart>
 <e-axes>
 <e-axis
 [labelBorder]='labelBorder'>
 </e-axis>
 </e-axes>
</ej-chart>

 <code>
 this.labelBorder: { width: 2, color: 'red' } | **Property:** *axis.labelBorder*

 <ejs-chart>
 <e-
 axes>
 <e-axis [labelBorder]='labelBorder'>
 </e-axis>
 </e-axes>
</ejs-chart>

 <code> this.labelBorder: { color: 'red', width: 2}

| Title of axis | **Property:** *axis.title*

 <ej-chart>
 <e-axes>
 <e-axis title='title'>

 </e-axis>
 </e-axes>
</ej-chart> | **Property:** *axis.title*

 <ejs-chart>
 <e-
 axes>
 <e-axis title='axis title'>
 </e-axis>
 </e-axes>
</ejs-chart> |

| skeleton of axis | **Property:** *axis.skeleton*

 <ej-chart>
 <e-axes>
 <e-axis
 skeleton='yMd'>
 </e-axis>
 </e-axes>
</ej-chart> | **Property:** *axis.skeleton*

 <ejs-chart>
 <e-axes>
 <e-axis skeleton='yMd'>
 </e-axis>
 </e-axes>

</ejs-chart>

| skeletonType of axis | **Property:** *axis.skeletonType*

 <ej-chart>
 <e-axes>
 <e-axis
 skeletonType='Date'>
 </e-axis>
 </e-axes>
</ej-chart> | **Property:** *axis.skeletonType*

 <ejs-chart>
 <e-axes>
 <e-axis skeletonType='Date'>
 </e-axis>
 </e-
 axes>
</ejs-chart> |

| Stripline of axis | **Property:** *axis.stripLines*

 <ej-chart>
 <e-axes>
 <e-axis>

 <e-stripLines>
 <e-stripLine>
 </e-stripLine>
 </e-stripLines>
 </e-axis>

 </e-axes>
</ej-chart> | **Property:** *axis.skeletonType*

 <ej-chart>
 <e-axes>

 <e-axis>
 <e-stripLines>
 <e-stripLine>
 </e-stripLine>
 </e-stripLines>
 </e-
 axis>
 </e-axes>
</ej-chart>

| Multilevel labels of axis | **Property:** *axis.multiLevelLabels*

 <ej-chart>
 <e-axes>
 <e-
 axis>
 <e-multiLevelLabels>
 <e-multiLevelLabel>
 </e-multiLevelLabel>
 </e-
 multiLevelLabels>
 </e-axis>
 </e-axes>
</ej-chart> | **Property:** *axis.multiLevelLabels*

 <ej-chart>
 <e-axes>
 <e-axis>
 <e-multilevellabels>
 <e-
 multilevellabel>
 </e-multilevellabel>
 </e-multilevellabels>
 </e-axis>
 </e-
 axes>
</ej-chart>

Series

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Series | **Property:** *series*

 <ej-chart>
<e-series-collection>
<e-series>
</e-series>
</e-series-collection>
</ej-chart> | **Property:** *series*

 <ejs-chart>
<e-series-collection>
<e-series>
</e-series>
</e-series-collection>
</ejs-chart> |

| bearFillColor | **Property:** *bearFillColor*

 <ej-chart>
<e-series-collection>
<e-series bearFillColor='red'>
</e-series>
</e-series-collection>
</ej-chart> | **Property:** *bearFillColor*

 <ejs-chart>
<e-series-collection>
<e-series bearFillColor='pink'>
</e-series>
</e-series-collection>
</ejs-chart> |

| boxPlotMode | **Property:** *boxPlotMode*

 <ej-chart>
<e-series-collection>
<e-series boxPlotMode='Inclusive'>
</e-series>
</e-series-collection>
</ej-chart> | **Property:** *bearFillColor*

 <ejs-chart>
<e-series-collection>
<e-series boxPlotMode='Inclusive'>
</e-series>
</e-series-collection>
</ejs-chart> |

| border | **Property:** *border*

 <ej-chart>
<e-series-collection>
<e-series [border]='border' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.border = { width: 2, color: 2 } | **Property:** *border*

 <ejs-chart>
<e-series-collection>
<e-series border='border'>
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.border = { width: 2, color: 'red' } |

| Minimum radius for bubble series | **Property:** *series.bubbleOptions.minRadius*

 <ej-chart>
<e-series-collection>
<e-series [bubbleOptions]='bubble' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.bubble = { minRadius: 3 } | **Property:** *series.MinRadius*

 <ejs-chart>
<e-series-collection>
<e-series [minRadius]='minimum' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.minimum = 2 |

| maximum radius for bubble series | **Property:** *series.bubbleOptions.maxRadius*

 <ej-chart>
<e-series-collection>
<e-series [bubbleOptions]='bubble' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.bubble = { maxRadius: 3 } | **Property:** *series.maxRadius*

 <ejs-chart>
<e-series-collection>
<e-series [maxRadius]='maximum' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.maximum = 2 |

| bullFillColor | **Property:** *bullFillColor*

 <ej-chart>
<e-series-collection>
<e-series bullFillColor='red'>
</e-series>
</e-series-collection>
</ej-chart> | **Property:** *bullFillColor*

 <ejs-chart>
<e-series-collection>
<e-series bullFillColor='pink'>
</e-series>
</e-series-collection>
</ejs-chart> |

| Cardinal spline tension | **Property:** *series.cardinalSplineTension*

 <ej-chart>
<e-series-collection>
<e-series [cardinalSplineTension]='spline' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.spline = 0.3 | **Property:** *series.maxRadius*

 <ejs-chart>
<e-series-collection>
<e-series [cardinalSplineTension]='spline' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.spline = 0.2 |

| Column spacing | **Property:** *series.columnSpacing*

 <ej-chart>
<e-series-collection>
<e-series [columnSpacing]='space' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.space = 0.3 | **Property:** *series.columnSpacing*

`</> <ej-chart>
<e-series-collection>
<e-series [columnSpacing]='space' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.space = 0.2 |`

| Column width | **Property:** `series.columnWidth` `

 <ej-chart>
<e-series-collection>
<e-series [columnWidth]='width' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.width = 0.3 |` **Property:** `series.columnWidth` `

 <ej-chart>
<e-series-collection>
<e-series [columnWidth]='width' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.width = 0.7 |`

| topLeft corner radius for rectangle series | **Property:** `series.cornerRadius.topLeft` `

 <ej-chart>
<e-series-collection>
<e-series [cornerRadius]='corner' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.corner = { topLeft: 5 } |` **Property:** `series.cornerRadius.topLeft` `

 <ej-chart>
<e-series-collection>
<e-series [cornerRadius]='radius' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.radius = { topLeft: 5 } |`

| topRight corner radius for rectangle series | **Property:** `series.cornerRadius.topRight` `

 <ej-chart>
<e-series-collection>
<e-series [cornerRadius]='corner' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.corner = { topRight: 5 } |` **Property:** `series.cornerRadius.topRight` `

 <ej-chart>
<e-series-collection>
<e-series [cornerRadius]='radius' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.radius = { topRight: 5 } |`

| bottomRight corner radius for rectangle series | **Property:** `series.cornerRadius.bottomRight` `

 <ej-chart>
<e-series-collection>
<e-series [cornerRadius]='corner' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.corner = { bottomRight: 5 } |` **Property:** `series.cornerRadius.bottomRight` `

 <ej-chart>
<e-series-collection>
<e-series [cornerRadius]='radius' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.radius = { bottomRight: 5 } |`

| bottomLeft corner radius for rectangle series | **Property:** `series.cornerRadius.bottomLeft` `

 <ej-chart>
<e-series-collection>
<e-series [cornerRadius]='corner' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.corner = { bottomLeft: 5 } |` **Property:** `series.cornerRadius.bottomLeft` `

 <ej-chart>
<e-series-collection>
<e-series [cornerRadius]='radius' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.radius = { bottomLeft: 5 } |`

| dashArray | **Property:** `series.dashArray` `

 <ej-chart>
<e-series-collection>
<e-series dashArray='10, 5'>
</e-series>
</e-series-collection>
</ej-chart> |` **Property:** `series.dashArray` `

 <ej-chart>
<e-series-collection>
<e-series dashArray='10, 5'>
</e-series>
</e-series-collection>
</ej-chart> |`

| dataSource for series | **Property:** `series.dataSource` `

 <ej-chart>
<e-series-collection>
<e-series [dataSource]='data' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.data = [] |` **Property:** `series.dataSource` `

 <ej-chart>
<e-series-collection>
<e-series [dataSource]='data' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.data = [] |`

| drawType for polar/radar series | **Property:** `series.drawType`

 <ej-chart>
<e-series-collection>
<e-series drawType='Spline' >
</e-series>
</e-series-collection>
</ej-chart>| **Property:** `series.drawType`

 <ejs-chart>
<e-series-collection>
<e-series [drawType]='data' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.data = 'Line' |

| Enable animation for series | **Property:** `series.enableAnimation`

 <ej-chart>
<e-series-collection>
<e-series [enableAnimation]='animation' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.animation = true | **Property:** `series.animation.enable`

 <ejs-chart>
<e-series-collection>
<e-series [animation]='animation' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.animation = { enable: true} |

| Duration animation for series | **Property:** `series.animationDuration`

 <ej-chart>
<e-series-collection>
<e-series [animationDuration]='animation' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.animation = 100 | **Property:** `series.animation.duration`

 <ejs-chart>
<e-series-collection>
<e-series [animation]='animation' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.animation = { enable: true} |

| Animation delay for series | **Property:** Not Applicable | **Property:** `series.animation.delay`

 <ejs-chart>
<e-series-collection>
<e-series [animation]='animation' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.animation = { delay: 100 } |

| Drag settings for series | **Property:** `series.animationDuration`

 <ej-chart>
<e-series-collection>
<e-series [dradSettings]='drag' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.drag = { mode: 'XY' } | **Property:** Not Applicable

| Customization of error bar | **Property:** `series.errorBar`

 <ej-chart>
<e-series-collection>
<e-series [errorbar]='errorBar' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.errorBar = {} | **Property:** `series.errorBar`

 <ejs-chart>
<e-series-collection>
<e-series [errorBar]='errorBar' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.errorBar = {} |

| isClosed | **Property:** `series.errorBar`

 <ej-chart>
<e-series-collection>
<e-series [errorbar]='errorBar' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.errorBar = {} | **Property:** `series.errorBar`

 <ejs-chart>
<e-series-collection>
<e-series [errorBar]='errorBar' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.errorBar = {} |

| enables Stacking of series | **Property:** `series.isStacking`

 <ej-chart>
<e-series-collection>
<e-series [isStacking]='isStacking' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.isStacking = true | **Property:** `series.errorBar`

 <ejs-chart>
<e-series-collection>
<e-series [isStacking]='isStacking' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.isStacking = true |

| Line cap of series | **Property:** `series.lineCap`

 <ej-chart>
<e-series-collection>
<e-series [lineCap]='lineCap' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.lineCap = 'butt' | **Property:** Not Applicable |

| Line join of series | **Property:** `series.lineJoin`

 <ej-chart>
<e-series-collection>
<e-series [lineJoin]='lineJoin' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.lineJoin = 'round' | **Property:** Not Applicable |

| Opacity of series | **Property:** `series.opacity`

 <ej-chart>
<e-series-collection>
<e-series [opacity]='opacity' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.opacity = 0.6 | **Property:** `series.opacity`

 <ejs-chart>
<e-series-collection>
<e-series [opacity]='opacity' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.opacity = true |

| Outlier settings of series | **Property:** `series.lineJoin`

 <ej-chart>
<e-series-collection>
<e-series [outLierSettings]='outlier' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.outLier = { } | **Property:** Not Applicable |

| Line join of series | **Property:** `series.lineJoin`

 <ej-chart>
<e-series-collection>
<e-series [lineJoin]='lineJoin' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.lineJoin = 'round' | **Property:** Not Applicable |

| Palette of series | **Property:** `series.palette`

 <ej-chart>
<e-series-collection>
<e-series [palette]='opacity' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.palette = [] | Not applicable

| Positive fill color for waterfall series | **Property:** `series.positiveColor`

 <ej-chart>
<e-series-collection>
<e-series [positiveColor]='positiveColor' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.positiveColor = 'red' | **Property:** `series.positiveColor`

 <ejs-chart>
<e-series-collection>
<e-series [positiveColor]='positiveColor' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.positiveColor = 'red' |

| To show average value in box and whisker series | **Property:** `series.showMedian`

 <ej-chart>
<e-series-collection>
<e-series [showMedian]='showMedian' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.showMedian = 'true' | **Property:** `series.showMean`

 <ejs-chart>
<e-series-collection>
<e-series [showMean]='showMean' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.showMean = true |

| Stacking group for stacking series | **Property:** `series.showMedian`

 <ej-chart>
<e-series-collection>
<e-series [stackingGroup]='stackingGroup' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.stackingGroup = 'copper' | **Property:** `series.stackingGroup`

 <ejs-chart>
<e-series-collection>
<e-series [stackingGroup]='stackingGroup' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.stackingGroup = 'copper' |

| Type of series | **Property:** `series.type`

 <ej-chart>
<e-series-collection>
<e-series [type]='type' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.type = 'Line' | **Property:** `series.type`

 <ejs-chart>
<e-series-collection>
<e-series [type]='type' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.type = 'Line' |

| Visibility of series | **Property:** `series.visibility`

 <ej-chart>
<e-series-collection>
<e-series [visibility]='visibility' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.visibility = true | **Property:** `series.visible`

 <ejs-chart>
<e-series-collection>
<e-series [visible]='visible' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.visible = true |

| Visibility on legend click | **Property:** `series.visibilityOnLegend`

 <ej-chart>
<e-series-collection>
<e-series [visibilityOnLegend]='visibility' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.visibility = true | **Property:** `legendSettings.toggleVisibility`

 <ejs-chart [legendSettings]='legend'>
<e-series-collection>
<e-series [visible]='visible' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.legend = { toggleVisibility = true } |

| Spline Type of series | **Property:** `series.splineType`

 <ej-chart>
<e-series-collection>
<e-series [splineType]='splineType' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.splineType = 'Monotonic' | **Property:** `series.splineType`

 <ejs-chart>
<e-series-collection>
<e-series [splineType]='splineType' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.splineType = 'Cardinal' |

| xAxisName of series | **Property:** `xAxisName`

 <ej-chart>
<e-series-collection>
<e-series xAxisName='xaxis'>
</e-series>
</e-series-collection>
</ej-chart> | **Property:** `xAxisName`

 <ejs-chart>
<e-series-collection>
<e-series xAxisName='axis'>
</e-series>
</e-series-collection>
</ejs-chart> |

| yAxisName of series | **Property:** `yAxisName`

 <ej-chart>
<e-series-collection>
<e-series yAxisName='xaxis'>
</e-series>
</e-series-collection>
</ej-chart> | **Property:** `yAxisName`

 <ejs-chart>
<e-series-collection>
<e-series yAxisName='axis'>
</e-series>
</e-series-collection>
</ejs-chart> |

| yAxisName of series | **Property:** `yAxisName`

 <ej-chart>
<e-series-collection>
<e-series yAxisName='xaxis'>
</e-series>
</e-series-collection>
</ej-chart> | **Property:** `yAxisName`

 <ejs-chart>
<e-series-collection>
<e-series yAxisName='axis'>
</e-series>
</e-series-collection>
</ejs-chart> |

| xName of series | **Property:** `xName`

 <ej-chart>
<e-series-collection>
<e-series xName='x'>
</e-series>
</e-series-collection>
</ej-chart> | **Property:** `xName`

 <ejs-chart>
<e-series-collection>
<e-series xName='x'>
</e-series>
</e-series-collection>
</ejs-chart> |

| yName of series | **Property:** `yName`

 <ej-chart>
<e-series-collection>
<e-series yName='y'>
</e-series>
</e-series-collection>
</ej-chart> | **Property:** `yName`

 <ejs-chart>
<e-series-collection>
<e-series yName='y'>
</e-series>
</e-series-collection>
</ejs-chart> |

| Name of the property in the datasource that contains high value for the series | **Property:** `high`

 <ej-chart>
<e-series-collection>
<e-series high='high'>
</e-series>
</e-series-collection>
</ej-chart> | **Property:** `high`

 <ejs-chart>
<e-series-

```
collection><br><e-series high="high"><br></e-series><br></e-series-collection><br></ejs-chart>
|
```

| Name of the property in the datasource that contains low value for the series | **Property:** *low*

 <ej-chart>
<e-series-collection>
<e-series low='low'>
</e-series>
</e-series-collection>
</ej-chart> | **Property:** *low*

 <ejs-chart>
<e-series-collection>
<e-series low="low">
</e-series>
</e-series-collection>
</ejs-chart> |

| Name of the property in the datasource that contains open value for the series | **Property:** *open*

 <ej-chart>
<e-series-collection>
<e-series open='open'>
</e-series>
</e-series-collection>
</ej-chart> | **Property:** *open*

 <ejs-chart>
<e-series-collection>
<e-series open="open">
</e-series>
</e-series-collection>
</ejs-chart> |

| Name of the property in the datasource that contains close value for the series | **Property:** *close*

 <ej-chart>
<e-series-collection>
<e-series close='close'>
</e-series>
</e-series-collection>
</ej-chart> | **Property:** *close*

 <ejs-chart>
<e-series-collection>
<e-series close="close">
</e-series>
</e-series-collection>
</ejs-chart> |

| Trendlines of series | **Property:** *trendlines*

 <ej-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ej-chart> | **Property:** *trendLines*

 <ejs-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ejs-chart>` |

| Marker settings in series | **Property:** *series.marker*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { } | **Property:** *series.marker*

 <ejs-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.marker = { } |

Marker

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Fill color in series | **Property:** *series.marker.fill*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { fill: 'red' } | **Property:** *series.marker.fill*

 <ejs-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.marker = { fill: 'red' } |

| Opacity of marker | **Property:** *series.marker.opacity*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { opacity: 0.4 } | **Property:** *series.marker.fill*

 <ejs-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.marker = { opacity: 0.4 } |

| shape of marker | **Property:** *series.marker.shape*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { shape: 'Circle' } | **Property:** *series.marker.fill*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.marker = { shape: 'Circle' } |

| imageUrl of marker | **Property:** *series.marker.imageUrl*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { imageUrl: './src.png' } | **Property:** *series.marker.fill*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.marker = { './src.png' } |

| border of marker | **Property:** *series.marker.border*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { border: { color: 'red', width: 2 } } | **Property:** *series.marker.fill*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.marker = { border:{ color: 'red', width: 2}} |

| Height of marker | **Property:** *series.marker.size.height*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { size: { height: 20 } } | **Property:** *series.marker.height*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.marker = { height: 20 } |

| width of marker | **Property:** *series.marker.size.width*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { size: { width: 20 } } | **Property:** *series.marker.width*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.marker = { width: 20 } |

| DataLabel of marker | **Property:** *series.marker.dataLabel*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { dataLabel: { visible: true } } | **Property:** *series.marker.dataLabel*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.marker = { dataLabel: { visible: true } } |

| TextMapping name of datalabel | **Property:** *marker.dataLabel.textMappingName*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { dataLabel: { textMappingName: 'name' } } | **Property:** *marker.dataLabel.name*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.marker = { dataLabel: { name: 'data' } } |

| Fill color of datalabel | **Property:** *marker.dataLabel.fill*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { dataLabel: { fill: 'red' } } | **Property:** *marker.dataLabel.fill*

 <ejs-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.marker = { dataLabel: { fill: 'red' } } |

| opacity color of datalabel| **Property:** *marker.dataLabel.opacity*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { dataLabel: { opacity: 0.6 } } | **Property:** *marker.dataLabel.opacity*

 <ejs-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.marker = { dataLabel: { opacity: 0.6 } } |

| Text position of datalabel| **Property:** *marker.dataLabel.textPosition*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { dataLabel: { textPosition: 'Top' } } | **Property:** *marker.dataLabel.position*

 <ejs-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.marker = { dataLabel: { position: 'Top' } } |

| Vertical alignment of datalabel| **Property:** *marker.dataLabel.verticalAlignment*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { dataLabel: { verticalAlignment: 'Near' } } | **Property:** *marker.dataLabel.alignment*

 <ejs-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.marker = { dataLabel: { alignment: 'far' } } |

| Border of datalabel| **Property:** *marker.dataLabel.border*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { dataLabel: { border: { color: 'red', width: 2 } } } | **Property:** *marker.dataLabel.border*

 <ejs-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.marker = { dataLabel: { border: { width: 2, color: 'blue' } } } |

| Margin of datalabel| **Property:** *marker.dataLabel.margin*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { dataLabel: { margin: { top: 10, left: 10, bottom: 10, right: 10 } } } | **Property:** *marker.dataLabel.margin*

 <ejs-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.marker = { dataLabel: { margin: { top: 10, left: 10, bottom: 10, right: 10 } } } |

| Offset of datalabel| **Property:** *marker.dataLabel.offset*

 <ej-chart>
<e-series-collection>
<e-series [border]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { dataLabel: { offset: 10 } } | **Property:** Not applicable |

| Text style of datalabel| **Property:** *marker.dataLabel.font*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { dataLabel: { font: { } } } | **Property:** *marker.dataLabel.font*

 <ejs-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-

series>
</e-series-collection>
</ejs-chart>

 <code>this.marker = { dataLabel: { font: { }} } |

| HTML template of datalabel| **Property:** *marker.dataLabel.template*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { dataLabel: { template:"" } } | **Property:** *marker.dataLabel.template*

 <ejs-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.marker = { dataLabel: { template: " " } } |

| Rounded corner x of datalabel| **Property:** Not Applicable **Property:** *marker.dataLabel.rx*

 <ejs-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.marker = { dataLabel: { rx: 10 } } |

| Rounded corner y of datalabel| **Property:** Not Applicable **Property:** *marker.dataLabel.ry*

 <ejs-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.marker = { dataLabel: { ry: 10 } } |

| Maximum width of datalabel| **Property:** *marker.dataLabel.maximumLabelWidth*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { dataLabel: { maximumLabelWidth:20 } } | **Property:** Not Applicable|

| Enabling wrap of datalabel| **Property:** *marker.dataLabel.enableWrap*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { dataLabel: { enableWrap:true } } | **Property:** Not Applicable|

| Show contrast color of datalabel| **Property:** *marker.dataLabel.showContrastColor*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { dataLabel: { showContrastColor:true } } | **Property:** Not Applicable|

| Show edge label of datalabel| **Property:** *marker.dataLabel.showEdgeLabel*

 <ej-chart>
<e-series-collection>
<e-series [marker]='marker'>
</e-series>
</e-series-collection>
</ej-chart>

 <code> this.marker = { dataLabel: { showEdgeLabel: true } } | **Property:** Not Applicable|

Error bar

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Type of error bar | **Property:** *series.errorBar.type*

 <ej-chart>
<e-series-collection>
<e-series [errorBar]='errorBar'>
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.errorBar = { type: 'StandardDeviation' } | **Property:** *series.errorBar.type*

 <ejs-chart>
<e-series-collection>
<e-series [errorBar]='errorBar'>
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.errorBar = { type: 'StandardDeviation' } |


```
| mode of error bar | Property: series.errorBar.mode <br/> <br/> <ej-chart> <br><e-series-  
collection><br><e-series [errorbar]='errorBar' ><br></e-series><br></e-series-  
collection><br></ej-chart> <br> <br> <code>this.errorBar = { mode: 'Horizontal' } | Property:  
series.errorBar.mode <br/> <br/> <ejs-chart> <br><e-series-collection><br><e-series  
[errorBar]='errorBar' ><br></e-series><br></e-series-collection><br></ejs-chart><br> <br>  
<code>this.errorBar = { mode: 'Horizontal' } |
```

| direction of error bar | **Property:** `series.errorBar.direction`

 <ej-chart>
<e-series-collection>
<e-series [errorBar]='errorBar' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.errorBar = { direction: 'Positive' } | **Property:** `series.errorBar.direction`

 <ejs-chart>
<e-series-collection>
<e-series [errorBar]='errorBar' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.errorBar = { direction: 'Positive' } |

| fill of error bar | **Property:** `series.errorBar.fill`

 <ej-chart>
 <e-series-collection>
 <e-series [errorBar]='errorBar' >
 </e-series>
 </e-series-collection>
 </ej-chart>

 <code>this.errorBar = { fill: 'red'} | **Property:** `series.errorBar.fill`

 <ejs-chart>
 <e-series-collection>
 <e-series [errorBar]='errorBar' >
 </e-series>
 </e-series-collection>
 </ejs-chart>

 <code>this.errorBar = { fill: 'red' } |

| width of error bar | **Property:** `series.errorBar.width`

 <ej-chart>
<e-series-collection>
<e-series [errorBar]='errorBar' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.errorBar = { width: 2 } | **Property:** `series.errorBar.width`

 <ejs-chart>
<e-series-collection>
<e-series [errorBar]='errorBar' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.errorBar = { width: 2 } |

| horizontalError of error bar | **Property:** `series.errorBar.horizontalError`

 <ej-chart>

<e-series-collection>
<e-series [errorbar]='errorBar' >
</e-series>
</e-series-
collection>
</ej-chart>

 <code>this.errorBar = { horizontalError: 2 } | **Property:**
`series.errorBar.horizontalError`

 <ejs-chart>
<e-series-collection>
<e-series
[errorBar]='errorBar' >
</e-series>
</e-series-collection>
</ejs-chart>

<code>this.errorBar = { horizontalError: 2 } |

| verticalError of error bar | **Property:** `series.errorBar.verticalError`

 <ej-chart>
 <e-series-collection>
 <e-series [errorBar]='errorBar' >
 </e-series>
 </e-series-collection>
 </ej-chart>

 <code>this.errorBar = { verticalError: 2 } | **Property:** `series.errorBar.verticalError`

 <ejs-chart>
 <e-series-collection>
 <e-series [errorBar]='errorBar' >
 </e-series>
 </e-series-collection>
 </ejs-chart>

 <code>this.errorBar = { verticalError: 2 } |

horizontalPositiveError of error bar | **Property:** `series.errorBar.horizontalPositiveError`

<ej-chart>
<e-series-collection>
<e-series [errorBar]='errorBar' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.errorBar = { horizontalPositiveError: 2 } |
Property: `series.errorBar.horizontalPositiveError`

 <ejs-chart>
<e-series-
collection>
<e-series [errorBar]='errorBar' >
</e-series>
</e-series-
collection>
</ejs-chart>

 <code>this.errorBar = { horizontalPositiveError: 2 } |

| horizontalNegativeError of error bar | **Property:** *series.errorBar.horizontalNegativeError*

 <ej-chart>
<e-series-collection>
<e-series [errorBar]='errorBar' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.errorBar = { horizontalNegativeError: 2 } |

Property: *series.errorBar.horizontalNegativeError*

 <ejs-chart>
<e-series-collection>
<e-series [errorBar]='errorBar' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.errorBar = { horizontalNegativeError: 2 } |

| verticalPositiveError of error bar | **Property:** *series.errorBar.verticalPositiveError*

 <ej-chart>
<e-series-collection>
<e-series [errorBar]='errorBar' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.errorBar = { verticalPositiveError: 2 } |

Property: *series.errorBar.verticalPositiveError*

 <ejs-chart>
<e-series-collection>
<e-series [errorBar]='errorBar' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.errorBar = { verticalPositiveError: 2 } |

| verticalNegativeError of error bar | **Property:** *series.errorBar.verticalNegativeError*

 <ej-chart>
<e-series-collection>
<e-series [errorBar]='errorBar' >
</e-series>
</e-series-collection>
</ej-chart>

 <code>this.errorBar = { verticalNegativeError: 2 } |

Property: *series.errorBar.verticalNegativeError*

 <ejs-chart>
<e-series-collection>
<e-series [errorBar]='errorBar' >
</e-series>
</e-series-collection>
</ejs-chart>

 <code>this.errorBar = { verticalNegativeError: 2 } |

Trendlines

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

|Trendline type of series| **Property:** *trendlines.type*

 <ej-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline type='Linear'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ej-chart> |

Property: *trendLines.type*

 <ejs-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline type='Linear'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ejs-chart> |

| Visiblity of Trendlines| **Property:** *trendlines.visible*

 <ej-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [visible]='visible'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ej-chart>

 <code>this.visible: true | **Property:** Not Applicable |

| Name of trendline| **Property:** *trendlines.name*

 <ej-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline name='linee1'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ej-chart> |

Property: *trendLines.name*

 <ejs-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline name='line1'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ejs-chart> |

| Period of trendline| **Property:** *trendlines.period*

 <ej-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [period]='period'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ej-chart>

 <code> this.period = 2 | **Property:** *trendLines.period*

 <ejs-chart>
<e-series-

collection>
<e-series>
<e-trendlines>
 <e-trendline [period]='period'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ejs-chart>

<code> this.period = 2 |

|polynomialOrder of trendline| **Property:** *trendlines.polynomialOrder*

 <ej-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [polynomialOrder]='polynomialOrder'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ej-chart>

<code> this.polynomialOrder = 2 |

Property: *trendLines.polynomialOrder*

 <ejs-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [polynomialOrder]='polynomialOrder'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ejs-chart>

<code> this.polynomialOrder = 2 |

|backwardForecast of trendline| **Property:** *trendlines.backwardForecast*

 <ej-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [backwardForecast]='backwardForecast'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ej-chart>

<code> this.backwardForecast = 2 |

Property: *trendLines.backwardForecast*

 <ejs-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [backwardForecast]='backwardForecast'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ejs-chart>

<code> this.backwardForecast = 2 |

|forwardForecast of trendline| **Property:** *trendlines.forwardForecast*

 <ej-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [forwardForecast]='forwardForecast'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ej-chart>

<code> this.forwardForecast = 2 |

Property: *trendLines.forwardForecast*

 <ejs-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [forwardForecast]='forwardForecast'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ejs-chart>

<code> this.forwardForecast = 2 |

|width of trendline| **Property:** *trendlines.width*

 <ej-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [width]='width'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ej-chart>

<code> this.width = 2 | **Property:** *trendLines.width*

 <ejs-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [width]='width'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ejs-chart>

<code> this.width = 2 |

|intercept of trendline| **Property:** *trendlines.intercept*

 <ej-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [intercept]='intercept'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ej-chart>

<code> this.intercept = 2 | **Property:** *trendLines.intercept*

 <ejs-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [intercept]='intercept'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ejs-chart>

<code> this.intercept = 2 |

| fill of trendline | **Property:** *trendlines.fill*

 <ej-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [fill]='fill'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ej-chart>

<code> this.fill = 'blue' |

Property: *trendLines.fill*

 <ejs-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [fill]='fill'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ejs-chart>

<code> this.fill = 'blue' |

| dashArray of trendline | **Property:** *trendlines.dashArray*

 <ej-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [dashArray]='dashArray'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ej-chart>

<code> this.dashArray = '10, 5' | **Property:** *trendLines.dashArray*

 <ejs-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [dashArray]='dashArray'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ejs-chart>

<code> this.dashArray = '10, 5' |

| legend shape of trendline | **Property:** Not Applicable | **Property:** *trendLines.legendShape*

 <ejs-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [legendShape]='Circle'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ejs-chart>

<code> this.circle = 'Circle' |

| Animation of trendline | **Property:** Not Applicable | **Property:** *trendLines.legendShape*

 <ejs-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [animation]='animation'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ejs-chart>

<code> this.animation = { } |

| Marker of trendline | **Property:** Not Applicable | **Property:** *trendLines.legendShape*

 <ejs-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [marker]='marker'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ejs-chart>

<code> this.marker = { } |

| tooltip of trendline | **Property:** *trendlines.dashArray*

 <ej-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [tooltip]='tooltip'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ej-chart>

<code> this.tooltip = { } | **Property:** *trendLines.enableTooltip*

 <ejs-chart>
<e-series-collection>
<e-series>
<e-trendlines>
 <e-trendline [enableTooltip]='enableTooltip'>
 </e-trendline>
 <e-trendlines>
 </e-series>
</e-series-collection>
</ejs-chart>

<code> this.enableTooltip = true |

Striplines

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Default | **Property:** *striplines*

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-striplines>
 <e-stripline>
 </e-stripline>
 <e-striplines>
 </e-axes>
</ej-chart> | **Property:** *striplines*

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-striplines>
 <e-stripline>
 </e-stripline>
 <e-striplines>
 </e-axes>
</ej-chart>

| Color of stripines | **Property:** *striplines*

 <ej-chart>
 <e-axes>
 <e-axis>

 </e-axis>
 <e-striplines>
 <e-stripline color='pink'>
 </e-stripline>
 <e-
 striplines>
 </e-axes>
</ej-chart> | **Property:** *striplines*

 <ej-chart>
 <e-
 axes>
 <e-axis>
 </e-axis>
 <e-striplines>
 <e-stripline color='pink'>
 </e-
 stripline>
 <e-striplines>
 </e-axes>
</ej-chart>

| Border of stripines | **Property:** *striplines.border*

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-striplines>
 <e-stripline [border]='border'>
 </e-stripline>
 <e-striplines>
 </e-axes>
 </ej-chart>

<code> this.border = { width: 2, color: 'red', opacity: 0.5 } | **Property:** *striplines.border*

 <ejs-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-striplines>
 <e-stripline [border]='border'>
 </e-stripline>
 <e-striplines>
 </e-axes>
</ejs-chart>

<code> this.border = { width: 2, color: 'red', opacity: 0.5 }

| Start of stripines | **Property:** *stripines.start*

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-stripines>
 <e-stripline [start]='start'>
 </e-stripline>
 <e-stripines>
 </e-axes>
</ej-chart>

<code> this.start = 2 | **Property:** *stripines.start*

 <ejs-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-stripines>
 <e-stripline [start]='start'>
 </e-stripline>
 <e-stripines>
 </e-axes>
</ejs-chart>

<code> this.start = 2

| end of stripines | **Property:** *striplines.end*

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-striplines>
 <e-stripline [end]='end'>
 </e-stripline>
 <e-striplines>
 </e-axes>
</ej-chart>

<code> this.end = 2 | **Property:** *striplines.end*

 <ejs-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-striplines>
 <e-stripline [end]='end'>
 </e-stripline>
 <e-striplines>
 </e-axes>
</ejs-chart>

<code> this.end = 2

```
| startFromAxis of striplines | Property: striplines.startFromAxis <br/> <br/> <ej-chart><br> <e-axes>
<br> <e-axis> <br> </e-axis> <br> <e-striplines> <br> <e-stripline
[startFromAxis]='startFromAxis'> <br> </e-stripline> <br> <e-striplines> <br> </e-axes> <br></ej-
chart> <br> <br><code> this.startFromAxis = true | Property: striplines.startFromAxis <br/> <br/> <ej-
chart><br> <e-axes> <br> <e-axis> <br> </e-axis> <br> <e-striplines> <br> <e-stripline
[startFromAxis]='startFromAxis'> <br> </e-stripline> <br> <e-striplines> <br> </e-axes>
<br></ej-chart> <br> <br><code> this.startFromAxis = false
```

| text of stripines | **Property:** *stripines.text*

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-stripines>
 <e-stripline [text]='text'>
 </e-stripline>
 <e-stripines>
 </e-axes>
</ej-chart>

<code> this.text = 'stripines' | **Property:** *stripines.text*

 <ejs-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-stripines>
 <e-stripline [text]='text'>
 </e-stripline>
 <e-stripines>
 </e-axes>
</ejs-chart>

<code> this.text = 'Stripines'

| size of stripines | **Property:** `striplines.width`

 `<ej-chart>`
 `<e-axes>`
 `<e-axis>`
 `</e-axis>`
 `<e-striplines>`
 `<e-stripline [width]='width'>`
 `</e-stripline>`
 `<e-striplines>`
 `</e-axes>`
 `</ej-chart>`

 `<code> this.width = 2` | **Property:** `striplines.size`

 `<ejs-chart>`
 `<e-axes>`
 `<e-axis>`
 `</e-axis>`
 `<e-striplines>`
 `<e-`

stripline [size]='size'>
 </e-stripline>
 <e-striplines>
 </e-axes>
 </ejs-chart>

<code> this.size = 2

| horizontal alignment of stripines | **Property:** *striplines.alignment*

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-striplines>
 <e-stripline [alignment]='alignment'>
 </e-stripline>
 <e-striplines>
 </e-axes>
 </ej-chart>

<code> this.horizontalAlignment = 'near' | **Property:** *striplines.horizontalAlignment*

 <ejs-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-striplines>
 <e-stripline [horizontalAlignment]='horizontalAlignment'>
 </e-stripline>
 <e-striplines>
 </e-axes>
 </ejs-chart>

<code> this.horizontalAlignment = 'near' |

| vertical alignment of stripines | **Property:** Not Applicable | **Property:** *striplines.horizontalAlignment*

 <ejs-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-striplines>
 <e-stripline [horizontalAlignment]='horizontalAlignment'>
 </e-stripline>
 <e-striplines>
 </e-axes>
 </ejs-chart>

<code> this.horizontalAlignment = 'near' |

| zIndex of stripines | **Property:** *striplines.zIndex*

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-striplines>
 <e-stripline [zIndex]='Over'>
 </e-stripline>
 <e-striplines>
 </e-axes>
 </ej-chart>

<code> this.zIndex = 'striplines' | **Property:** *striplines.zIndex*

 <ejs-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-striplines>
 <e-stripline [zIndex]='zIndex'>
 </e-stripline>
 <e-striplines>
 </e-axes>
 </ejs-chart>

<code> this.zIndex = 'Behind'

| Font style of stripines text | **Property:** *striplines.font*

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-striplines>
 <e-stripline [font]='font'>
 </e-stripline>
 <e-striplines>
 </e-axes>
 </ej-chart>

<code> this.font = {} | **Property:** *striplines.textStyle*

 <ejs-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-striplines>
 <e-stripline [textStyle]='text'>
 </e-stripline>
 <e-striplines>
 </e-axes>
 </ejs-chart>

<code> this.text = { }

Multilevel labels

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Default | **Property:** *multilevellabels*

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-multilevellabels>
 <e-multilevellabel>
 </e-multilevellabel>
 <e-multilevellabels>
 </e-axes>
 </ej-chart> | **Property:** *multilevellabels*

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-multilevellabels>
 <e-multilevellabel>
 </e-multilevellabel>
 <e-multilevellabels>
 </e-axes>
 </ej-chart>

| Text alignment in labels | **Property:** *textAlignment*

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-multilevellabels>
 <e-multilevellabel textAlignment='near'>
 </e-multilevellabel>
 <e-multilevellabels>
 </e-axes>
 </ej-chart> | **Property:** *multilevellabels*

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-multilevellabels>
 <e-multilevellabel alignment='near'>
 </e-multilevellabel>
 <e-multilevellabels>
 </e-axes>
 </ej-chart>

| Text overflow in labels | **Property:** `textOverflow`

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-multilevellabels>
 <e-multilevellabel textOverflow='trim'>
 </e-multilevellabel>
 <e-multilevellabels>
 </e-axes>
</ej-chart> | **Property:** `overflow`

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-multilevellabels>
 <e-multilevellabel overflow='Trim'>
 </e-multilevellabel>
 <e-multilevellabels>
 </e-axes>
</ej-chart>

| Text of multilevel labels| **Property:** `text`

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-multilevellabels>
 <e-multilevellabel text='Texts'>
 </e-multilevellabel>
 <e-multilevellabels>
 </e-axes>
</ej-chart> | **Property:** `categories.text`

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-multilevellabels>
 <e-multilevellabel >
 <e-categories>
 <e-category text='text'>
 </e-category>
 </e-categories> </e-multilevellabel>
 <e-multilevellabels>
 </e-axes>
</ej-chart>

| Start of multilevel labels| **Property:** `start`

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-multilevellabels>
 <e-multilevellabel start='start'>
 </e-multilevellabel>
 <e-multilevellabels>
 </e-axes>
</ej-chart>

 this.start = 20 | **Property:** `categories.start`

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-multilevellabels>
 <e-multilevellabel >
 <e-categories>
 <e-category start='start'>
 </e-category>
 </e-categories> </e-multilevellabel>
 <e-multilevellabels>
 </e-axes>
</ej-chart>

 this.start = 20|

| End of multilevel labels| **Property:** `end`

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-multilevellabels>
 <e-multilevellabel start='start'>
 </e-multilevellabel>
 <e-multilevellabels>
 </e-axes>
</ej-chart>

 this.end = 20 | **Property:** `categories.end`

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-multilevellabels>
 <e-multilevellabel >
 <e-categories>
 <e-category end='end'>
 </e-category>
 </e-categories> </e-multilevellabel>
 <e-multilevellabels>
 </e-axes>
</ej-chart>

 this.end = 20|

| Maximum Label width | **Property:** `maximumLabelWidth`

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-multilevellabels>
 <e-multilevellabel [maximumLabelWidth]='width'>
 </e-multilevellabel>
 <e-multilevellabels>
 </e-axes>
</ej-chart>

 this.width = 16 | **Property:** `maximumLabelWidth`

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-multilevellabels>
 <e-multilevellabel [maximumLabelWidth]='width'>
 </e-multilevellabel>
 <e-multilevellabels>
 </e-axes>
</ej-chart>

 <code> this.width = 20;

| Level of labels | **Property:** `level`

 <ej-chart>
 <e-axes>
 <e-axis>
 </e-axis>
 <e-multilevellabels>
 <e-multilevellabel [level]='level'>
 </e-multilevellabel>
 <e-multilevellabels>
 </e-axes>
</ej-chart>

 this.level = 3 | **Property:** Not Applicable

Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Animation | **Property:** `chart.animate`

 <ej-chart (animate)='animation'>
</ej-chart>

 <code> this.animate(args) { } | **Property:** Not Applicable

| Redraw | **Property:** `chart.redraw`

 <ej-chart (redraw)='redraw'>
</ej-chart>

 <code> this.redraw(args) { } | **Property:** `chart.refresh`

 <ej-chart (refresh)='refresh'>
</ej-chart>

 <code> this.refresh(args) { }

| print | **Property:** `chart.print`

 <ej-chart (print)='print'>
</ej-chart>

 <code> this.print(args) { } | **Property:** `chart.print`

 <ej-chart (print)='print'>
</ej-chart>

 <code> this.print(args) { }

| export | **Property:** `chart.export`

 <ej-chart (print)='print'>
</ej-chart>

 <code> this.print(args) { } | **Property:** `chart.export`

 <ej-chart (export)='export'>
</ej-chart>

 <code> this.export(args) { }

| add series | **Property:** Not Applicable | **Property:** `chart.export`

 <ej-chart (addSeries)='addSeries'>
</ej-chart>

 <code> this.addSeries(args) { }

| remove series | **Property:** Not Applicable | **Property:** `chart.export`

 <ej-chart (removeSeries)='removeSeries'>
</ej-chart>

 <code> this.removeSeries(args) { }

Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Fires on annotation click | **Property:** `chart.annotationClick`

 <ej-chart (annotationClick)='annotationClick(\$event)'>
</ej-chart>

 <code> this.annotationClick(args) { } | **Property:** Not Applicable

| Fires on after animation | **Property:** `chart.animationComplete`

 <ej-chart (animationComplete)='animationComplete(\$event)'>
</ej-chart>

 <code> this.animationComplete(args) { } | **Property:** `chart.animationComplete`

 <ej-chart (refresh)='refresh'>
</ej-chart>

 <code> this.refresh(args) { }

| Fires on axis label click | **Property:** `chart.axisLabelClick`

 <ej-chart (axisLabelClick)='axisLabelClick(\$event)'>
</ej-chart>

 <code> this.axisLabelClick(args) { } | **Property:** Not Applicable

| Fires before axis label rendering | **Property:** `chart.axisLabelRendering`

 <ej-chart (axisLabelRendering)='axisLabelRendering(\$event)'>
</ej-chart>

 <code> this.axisLabelRender(args) { } | **Property:** `chart.axisLabelRender`

 <ej-chart (axisLabelRender)='axisLabelRender(\$event)'>
</ej-chart>

 <code> this.axisLabelRender(args) { }

| Fires on axis label mouse move | **Property:** `chart.axisLabelMouseMove`

 <ej-chart (axisLabelMouseMove)='axisLabelMouseMove(\$event)'>
</ej-chart>

 <code> this.axisLabelMouseMove(args) { } | **Property:** Not Applicable

| Fires on axis label initialize | **Property:** `chart.axisLabelMouseMove`

 <ej-chart (axisLabelInitialize)='axisLabelInitialize(\$event)'>
</ej-chart>

 <code> this.axisLabelInitialize(args) { } | **Property:** Not Applicable

| Fires before axis range calculate | **Property:** *chart.axisRangeCalculate*

 <ej-chart (axisRangeCalculate)=‘axisRangeCalculate(\$event)’>
</ej-chart>

 <code> this.axisRangeCalculate(args) { } | **Property:** *chart.axisRangeCalculated*

 <ejs-chart (axisRangeCalculated)=‘axisRangeCalculated(\$event)’>
</ejs-chart>

 <code> this.axisRangeCalculated(args) { }

| Fires on axis title rendering | **Property:** *chart.axisTitleRendering*

 <ej-chart (axisTitleRendering)=‘axisTitleRendering(\$event)’>
</ej-chart>

 <code> this.axisTitleRendering(args) { } | **Property:** Not Applicable

| Fires on after chart resize | **Property:** *chart.afterResize*

 <ej-chart (afterResize)=‘afterResize(\$event)’>
</ej-chart>

 <code> this.afterResize(args) { } | **Property:** Not Applicable

| Fires before resize | **Property:** *chart.beforeResize*

 <ej-chart (beforeResize)=‘beforeResize(\$event)’>
</ej-chart>

 <code> this.beforeResize(args) { } | **Property:** *chart.resized*

 <ejs-chart (resized)=‘resized(\$event)’>
</ejs-chart>

 <code> this.resized(args) { }

| Fires chart click | **Property:** *chart.chartClick*

 <ej-chart (chartClick)=‘chartClick(\$event)’>
</ej-chart>

 <code> this.chartClick(args) { } | **Property:** *chart.chartMouseClicked*

 <ejs-chart (chartMouseClicked)=‘chartMouseClicked(\$event)’>
</ejs-chart>

 <code> this.chartMouseClicked(args) { }

| Fires chart mouse leave | **Property:** *chart.chartMouseLeave*

 <ej-chart (chartMouseLeave)=‘chartMouseLeave(\$event)’>
</ej-chart>

 <code> this.chartMouseLeave(args) { } | **Property:** *chart.chartMouseClicked*

 <ejs-chart (chartMouseLeave)=‘chartMouseLeave(\$event)’>
</ejs-chart>

 <code> this.chartMouseLeave(args) { }

| Fires on double click | **Property:** *chart.chartDoubleClick*

 <ej-chart (chartDoubleClick)=‘chartDoubleClick(\$event)’>
</ej-chart>

 <code> this.chartDoubleClick(args) { } | **Property:** Not Applicable

| Fires chart mouse up | **Property:** Not Applicable | **Property:** *chart.chartmouseUp*

 <ejs-chart (chartmouseUp)=‘chartmouseUp(\$event)’>
</ejs-chart>

 <code> this.chartmouseUp(args) { }

| Fires on chart mouse up | **Property:** Not Applicable | **Property:** *chart.chartmouseDown*

 <ejs-chart (chartmouseDown)=‘chartmouseDown(\$event)’>
</ejs-chart>

 <code> this.chartmouseDown(args) { }

| Fires during calculation of area bounds | **Property:** *chart.chartAreaBoundsCalculate*

 <ej-chart (chartAreaBoundsCalculate)=‘chartAreaBoundsCalculate(\$event)’>
</ej-chart>

 <code> this.chartAreaBoundsCalculate(args) { } | **Property:** Not Applicable

| Fires on drag start | **Property:** *chart.dragStart*

 <ej-chart (dragStart)=‘dragStart(\$event)’>
</ej-chart>

 <code> this.dragStart(args) { } | **Property:** Not Applicable

| Fires on dragging| **Property:** *chart.dragging*

 <ej-chart
(dragging)='dragging(\$event)'>
</ej-chart>

 <code> this.dragging(args) { } | **Property:**
Not Applicable

| Fires on drag end| **Property:** *chart.dragEnd*

 <ej-chart
(dragEnd)='dragEnd(\$event)'>
</ej-chart>

 <code> this.dragEnd(args) { } | **Property:**
chart.dragComplete

 <ejs-chart (dragComplete)='dragComplete(\$event)'>
</ejs-
chart>

 <code> this.dragComplete(args) { }

| Fires after chart destroyed| **Property:** *chart.destroy*

 <ej-chart
(destroy)='destroy(\$event)'>
</ej-chart>

 <code> this.destroy(args) { } | **Property:** Not
Applicable

| Fires on chart created| **Property:** *chart.create*

 <ej-chart
(create)='create(\$event)'>
</ej-chart>

 <code> this.create(args) { } | **Property:**
chart.load

 <ejs-chart (load)='load(\$event)'>
</ejs-chart>

 <code>
this.load(args) { }

| Fires on data labeltext render| **Property:** *chart.displayTextRendering*

 <ej-chart
(displayTextRendering)='displayTextRendering(\$event)'>
</ej-chart>

 <code>
this.displayTextRendering(args) { } | **Property:** *chart.textRender*

 <ejs-chart
(textRender)='textRender(\$event)'>
</ejs-chart>

 <code> this.textRender(args) { }

| Fires on errorbar render| **Property:** *chart.errorbarRendering*

 <ej-chart
(errorBarRendering)='errorBarRendering(\$event)'>
</ej-chart>

 <code>
this.errorBarRendering(args) { } | **Property:** Not Applicable.

| Fires on legend bound calculate| **Property:** *chart.errorbarRendering*

 <ej-chart
(legendBoundsCalculate)='legendBoundsCalculate(\$event)'>
</ej-chart>

 <code>
this.legendBoundsCalculate(args) { } | **Property:** Not Applicable.

| Fires on legend item click| **Property:** *chart.legendItemClick*

 <ej-chart
(legendItemClick)='legendItemClick(\$event)'>
</ej-chart>

 <code>
this.legendItemClick(args) { } | **Property:** Not Applicable.

| Fires on legend item mouse move| **Property:** *chart.legendItemMouseMove*

 <ej-chart
(legendItemMouseMove)='legendItemMouseMove(\$event)'>
</ej-chart>

 <code>
this.legendItemMouseMove(args) { } | **Property:** Not Applicable.

| Fires on legend item render| **Property:** *chart.legendItemRendering*

 <ej-chart
(legendItemRendering)='legendItemMouseMove(\$event)'>
</ej-chart>

 <code>
this.legendItemMouseMove(args) { } | **Property:** Not Applicable.

| Fires on multilevel label render| **Property:** *chart.multiLevelLabelRendering*

 <ej-chart
(multiLevelLabelRendering)='multiLevelLabelRendering(\$event)'>
</ej-chart>

<code> this.multiLevelLabelRendering(args) { } | **Property:** *chart.axisMultiLabelRender*

<ejs-chart (axisMultiLabelRender)='axisMultiLabelRender(\$event)'>
</ejs-chart>

<code> this.axisMultiLabelRender(args) { }

| Fires on point click| **Property:** *chart.pointRegionClick*

 <ej-chart
(pointRegionClick)='pointRegionClick(\$event)'>
</ej-chart>

 <code>

this.pointRegionClick(args) {} | **Property:** *chart.pointClick*

 <ej-chart
 (pointClick)='pointClick(\$event)'>
</ej-chart>

 <code> this.pointClick(args) {}

| Fires on multilevel label render | **Property:** *chart.pointRegionMouseMove*

 <ej-chart
 (pointRegionMouseMove)='pointRegionMouseMove(\$event)'>
</ej-chart>

 <code>
 this.pointRegionMouseMove(args) {} | **Property:** *chart.pointMouseMove*

 <ej-chart
 (pointMouseMove)='pointMouseMove(\$event)'>
</ej-chart>

 <code>
 this.pointMouseMove(args) {}

| Fires on pre render | **Property:** *chart.preRender*

 <ej-chart
 (preRender)='preRender(\$event)'>
</ej-chart>

 <code> this.preRender(args) {} |
Property: Not Applicable.

| Fires on point render | **Property:** Not Applicable. | **Property:** *chart.pointRender*

 <ejss-
 chart (pointRender)='pointRender(\$event)'>
</ejss-chart>

 <code>
 this.pointRender(args) {}

| Fires on range selected | **Property:** *chart.rangeSelected*

 <ej-chart
 (rangeSelected)='rangeSelected(\$event)'>
</ej-chart>

 <code>
 this.rangeSelected(args) {} | **Property:** Not Applicable.

| Fires on series render | **Property:** *chart.seriesRendering*

 <ej-chart
 (seriesRendering)='seriesRendering(\$event)'>
</ej-chart>

 <code>
 this.seriesRendering(args) {} | **Property:** *chart.seriesRender*

 <ej-chart
 (seriesRender)='seriesRender(\$event)'>
</ej-chart>

 <code> this.seriesRender(args) {}

| Fires on trendLine render | **Property:** *chart.trendLineRendering*

 <ej-chart
 (trendLineRendering)='trendLineRendering(\$event)'>
</ej-chart>

 <code>
 this.trendLineRendering(args) {} | **Property:** Not Applicable.

| Fires on title render | **Property:** *chart.titleRendering*

 <ej-chart
 (titleRendering)='titleRendering(\$event)'>
</ej-chart>

 <code>
 this.titleRendering(args) {} | **Property:** Not Applicable.

| Fires on sub title render | **Property:** *chart.subTitleRendering*

 <ej-chart
 (subTitleRendering)='subTitleRendering(\$event)'>
</ej-chart>

 <code>
 this.subTitleRendering(args) {} | **Property:** Not Applicable.

| Fires before tooltip render | **Property:** *chart.tooltipInitilize*

 <ej-chart
 (tooltipInitilize)='tooltipInitilize(\$event)'>
</ej-chart>

 <code> this.tooltipInitilize(args)
 {} | **Property:** *chart.tooltipRender*

 <ej-chart
 (tooltipRender)='tooltipRender(\$event)'>
</ej-chart>

 <code> this.tooltipRender(args)
 {}

| Fires on cross hair tooltip render | **Property:** *chart.trackAxisTooltip*

 <ej-chart
 (trackAxisTooltip)='trackAxisTooltip(\$event)'>
</ej-chart>

 <code>
 this.trackAxisTooltip(args) {} | **Property:** Not Applicable.

| Fires on before track ball rendering | **Property:** *chart.trackTooltip*

 <ej-chart
 (trackTooltip)='trackTooltip(\$event)'>
</ej-chart>

 <code> this.trackTooltip(args) {} |
Property: Not Applicable.

| Fires when scroll start| **Property:** `chart.scrollStart`

 <ej-chart
(scrollStart)=`'scrollStart($event)'`>
</ej-chart>

 <code> this.scrollStart(args) { } |
Property: `chart.scrollStart`

 <ejs-chart (scrollStart)=`'scrollStart($event)'`>
</ejs-
chart>

 <code> this.scrollStart(args) { }

| Fires when scroll End| **Property:** `chart.scrollEnd`

 <ej-chart
(scrollEnd)=`'scrollEnd($event)'`>
</ej-chart>

 <code> this.scrollEnd(args) { } | **Property:**
`chart.scrollEnd`

 <ejs-chart (scrollEnd)=`'scrollEnd($event)'`>
</ejs-chart>

<code> this.scrollEnd(args) { }

| Fires when scroll Change| **Property:** `chart.scrollChange`

 <ej-chart
(scrollChange)=`'scrollChange($event)'`>
</ej-chart>

 <code> this.scrollChange(args) { }
| **Property:** `chart.scrollChange`

 <ejs-chart
(scrollChange)=`'scrollChange($event)'`>
</ejs-chart>

 <code> this.scrollChange(args) { }

| Fires when zoom complete| **Property:** `chart.zoomComplete`

 <ej-chart
(zoomComplete)=`'zoomComplete($event)'`>
</ej-chart>

 <code>
this.zoomComplete(args) { } | **Property:** `chart.scrollChange`

 <ejs-chart
(zoomComplete)=`'zoomComplete($event)'`>
</ejs-chart>

 <code>
this.zoomComplete(args) { }

How To

Live chart in Angular Chart component

You can update a chart with live data by using the set interval.

To update live data in a chart, follow the given steps:

Step 1:

Initialize the chart with series.

```
`javascript
import { Component } from '@angular/core';
@Component({
  selector: 'app-container',
  // specifies the template string for the Chart component
  template: <ejs-chart id="chart-container"></ejs-chart>
})
export class AppComponent {
  <e-series-collection>
  <e-series [dataSource]='data' type='Line' xName='x' yName='y'> </e-series>
</e-series-collection>
}
```

Step 2:

Update the data to series, and refresh the chart at specified interval by using the set interval.

To refresh the chart, invoke the `refresh` method.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { PieSeriesService, AccumulationTooltipService,
 AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {
  LineSeriesService, DateTimeService, DataLabelService,
  StackingColumnSeriesService, CategoryService,
  StepAreaSeriesService, SplineSeriesService, ScrollBarService,
  ChartAnnotationService, LegendService, TooltipService, StripLineService,
  SelectionService, ScatterSeriesService, ZoomService,
  ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
} from '@syncfusion/ej2-angular-charts'
import { Component, ViewEncapsulation, ViewChild } from '@angular/core';
import { ILoadedEventArgs, ChartComponent } from '@syncfusion/ej2-angular-
charts';
import { getElement } from '@syncfusion/ej2-charts';
@Component({
  imports: [
    ChartModule, ChartAllModule, AccumulationChartAllModule,
    AccumulationChartModule
  ],
  providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
    DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
    LegendService, TooltipService, StripLineService,
    PieSeriesService, AccumulationTooltipService, ScrollBarService,
    AccumulationDataLabelService, SelectionService, ScatterSeriesService,
    AreaSeriesService, RangeAreaSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart #chart id='chart-container'
 [chartArea]='chartArea' [width]='width' align='center'
 [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
 (loaded)='loaded($event)'>
    <e-series-collection>
      <e-series [dataSource]='series1' type='Line' xName='x' yName='y'
width=2 [animation]='animation1'>
    </e-series>
    </e-series-collection>
  </ejs-chart>`,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  public series1: Object[] = [];
  public value: number = 10;
  public intervalId?: any;
  public setTimeoutValue?: number;
```

```

public i: number = 0;
//Initializing Primary Y Axis
public primaryYAxis: Object = {
    minimum:0,
    maximum: 50
};
@ViewChild('chart')
public chart?: ChartComponent;
public marker: Object = {
    visible: true
};
public animation1: Object = {
    enable: false
};
chartArea: any;
width: any;
primaryXAxis: any;
title: any;
public loaded(args: ILoadedEventArgs): void {
    this.setTimeoutValue = 100;
    this.intervalId = setInterval(
        () => {
            let i: number;
            if (getElement('chart-container') === null) {
                clearInterval(this.intervalId);
            } else {
                if (Math.random() > .5) {
                    if (this.value < 25) {
                        this.value += Math.random() * 2.0;
                    } else {
                        this.value -= 2.0;
                    }
                }
                this.i++;
                this.series1.push({ x: this.i, y: this.value });
                this.series1.shift();
                args.chart.series[0].dataSource = this.series1;
                args.chart.refresh();
            }
        },
        this.setTimeoutValue);
}
constructor() {
    for (; this.i < 100; this.i++) {
        if (Math.random() > .5) {
            if (this.value < 25) {
                this.value += Math.random() * 2.0;
            } else {
                this.value -= 2.0;
            }
        }
        this.series1[this.i] = { x: this.i, y: this.value };
    }
};
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Prevent data label in Angular Chart component

To prevent the chart data label when the data value is 0, follow the given steps:

Step 1:

Get the point value and check whether the `args.point.y` value is zero or not by using the [textRender](#) event. If the value is zero, then set the `args.cancel` to true.

The output will appear as follows,

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { PieSeriesService, AccumulationTooltipService,
AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {
    LineSeriesService, DateTimeService, DataLabelService,
    StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ScrollBarService,
    ChartAnnotationService, LegendService, TooltipService, StripLineService,
    SelectionService, ScatterSeriesService, ZoomService,
    ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { ITextRenderEventArgs } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    ChartModule, ChartAllModule, AccumulationChartAllModule,
    AccumulationChartModule
  ],
  providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
    DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
    LegendService, TooltipService, StripLineService,
    PieSeriesService, AccumulationTooltipService, ScrollBarService,
    AccumulationDataLabelService, SelectionService, ScatterSeriesService,
    AreaSeriesService, RangeAreaSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
(textRender)='textRender($event)'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='India' width=2 [marker]='marker'></e-series>
    </e-series-collection>
```

```

</ejs-chart>`
}))
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  public primaryYAxis?: Object;
  public textRender(args: ITextRenderEventArgs): void {
    if (args.text === '0') {
      args.cancel = args.point.y === 0;
    }
  };
  ngOnInit(): void {
    this.chartData = [
      { x: new Date(2005, 0, 1), y: 21 }, { x: new Date(2006, 0, 1),
y: 24 },
      { x: new Date(2007, 0, 1), y: 0 }, { x: new Date(2008, 0, 1),
y: 38 },
      { x: new Date(2009, 0, 1), y: 54 }, { x: new Date(2010, 0, 1),
y: 57 },
    ];
    this.primaryXAxis = {
      title: 'Year',
      valueType: 'DateTime'
    };
    this.primaryYAxis = {
      title: 'Efficiency',
    };
    this.marker = { visible: true, width: 10, height: 10, dataLabel: {
visible: true}};
    this.title = 'Inflation - Consumer Price';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Tool tip format in Angular Chart component

Using [tooltipRender](#) event, you can able to format the

datetime value instead of rendered value.

To format the datetime value, please follow the steps below

Step 1:

By using [tooltipRender](#) event we can able to get the current point x value. Using this value to format the tooltip by using `formatDate` method.

The output will appear as follows,

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { PieSeriesService, AccumulationTooltipService,
 AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {
  LineSeriesService, DateTimeService, DataLabelService,
  StackingColumnSeriesService, CategoryService,
  StepAreaSeriesService, SplineSeriesService, ScrollBarService,
  ChartAnnotationService, LegendService, TooltipService, StripLineService,
  SelectionService, ScatterSeriesService, ZoomService,
  ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { ITooltipRenderEventArgs } from '@syncfusion/ej2-angular-charts';
import { Internationalization } from '@syncfusion/ej2-base';
@Component({
  imports: [
    ChartModule, ChartAllModule, AccumulationChartAllModule,
    AccumulationChartModule
  ],
  providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
    DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
    LegendService, TooltipService, StripLineService,
    PieSeriesService, AccumulationTooltipService, ScrollBarService,
    AccumulationDataLabelService, SelectionService, ScatterSeriesService,
    AreaSeriesService, RangeAreaSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'
[tooltip]='tooltip' (tooltipRender) = 'tooltipRender($event)'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='India' width=2 [marker]='marker'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  public tooltip?: Object;
  public primaryYAxis?: Object;
  public tooltipRender(args: ITooltipRenderEventArgs | any): void {
    let intl: Internationalization = new Internationalization();
    let formattedString: string = intl.formatDate(new
Date((args.point.x).toString()), { skeleton: 'MMMd' });
    args.text = formattedString + ':' + args.text.split(':')[1];
  };
  ngOnInit(): void {

```

```

        this.chartData = [
            { x: new Date(2005, 0, 1), y: 21 }, { x: new Date(2006, 0, 1),
y: 24 },
            { x: new Date(2007, 0, 1), y: 30 }, { x: new Date(2008, 0, 1),
y: 38 },
            { x: new Date(2009, 0, 1), y: 54 }, { x: new Date(2010, 0, 1),
y: 57 },
        ];
        this.primaryXAxis = {
            title: 'Year',
            valueType: 'DateTime'
        };
        this.primaryYAxis = {
            title: 'Efficiency',
        };
        this.marker = { visible: true, width: 10, height: 10, dataLabel: {
visible: true}};
        this.title = 'Inflation - Consumer Price';
        this.tooltip = {enable: true}
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Add series in Angular Chart component

You can add or remove the chart series dynamically by using the `addSeries` or `removeSeries` method.

To add or remove the series dynamically, follow the given steps:

Step 1:

To add a new series to chart dynamically, pass the series value to the `addSeries` method.

To remove the new series from chart dynamically, pass the series index to the `removeSeries` method.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CategoryService, ColumnSeriesService, ExportService, LegendService,
DataLabelService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { ChartComponent } from '@syncfusion/ej2-angular-charts';
@Component({
    imports: [
        ChartModule, ButtonModule
    ],
    providers: [ CategoryService, ColumnSeriesService, ExportService,
LegendService, DataLabelService],

```

```

standalone: true,
  selector: 'app-container',
  template: `<ejs-chart #chart id='chartcontainer'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
    [title]='title' >
    <e-series-collection>
      <e-series [dataSource]='data' type='Column' xName='x'
yName='y' > </e-series>
    </e-series-collection>
  </ejs-chart>
  <button ejs-button class='e-flat' (click)='add()'>add </button>
  <button ejs-button class='e-flat' (click)='remove()'>remove </button>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public primaryYAxis?: Object;
  public data?: Object[];
  @ViewChild('chart')
  public chart?: ChartComponent;
  ngOnInit(): void {
    this.data = [{ x: 'John', y: 10000 }, { x: 'Jake', y: 12000 }, { x:
'Peter', y: 18000 },
      { x: 'James', y: 11000 }, { x: 'Mary', y: 9700 }];
    this.primaryXAxis = {
      title: 'Manager',
      valueType: 'Category'
    };
    this.primaryYAxis = {
      title: 'Sales',
      minimum: 0,
      maximum: 20000
    };
    this.title = 'Sales Comparision';
  }
  add() {
    (this.chart as ChartComponent).addSeries([
      {
        type: 'Column',
        dataSource: [
          { x: 'John', y: 11000 }, { x: 'Jake', y: 16000 }, { x: 'Peter',
y: 19000 },
          { x: 'James', y: 12000 }, { x: 'Mary', y: 10700 }],
        xName: 'x', width: 2,
        yName: 'y'
      }
    ]);
  }
  remove() {
    (this.chart as ChartComponent).removeSeries(1);
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';

```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Points customization in Angular Chart component

You can customize the series points by using the `pointColorMapping` property.

To customize the series point colors, follow the given steps:

Step 1:

Customize the point colors to set the color value by using the `pointColorMapping` property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { ColumnSeriesService, CategoryService, DataLabelService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ ColumnSeriesService, CategoryService, DataLabelService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-series-collection>
    <e-series [dataSource]='data' type='Column' xName='x' yName='y'
name='Tiger' width=2 [marker]='marker' [cornerRadius]='radius'
    [pointColorMapping]='pointColorMapping'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public title?: string;
  public primaryYAxis?: Object;
  public data?: Object[];
  public pointColorMapping?: string;
  public radius?: Object;
  marker: any;
  ngOnInit(): void {
    this.data = [
      { x: 'BGD', y: 106, text: 'Bangladesh', color:
'url(#chess)' },
      { x: 'BTN', y: 103, text: 'Bhutn', color: 'url(#cross)' },
      { x: 'NPL', y: 198, text: 'Nepal', color: 'url(#circle)' },
      { x: 'THA', y: 189, text: 'Thiland', color:
'url(#rectangle)' },
      { x: 'MYS', y: 250, text: 'Malaysia', color:
'url(#line)' }
    ];
    this.primaryXAxis = {
```

```

        valueType: 'Category', interval: 1, majorGridLines: { width: 0 }
    };
    this.primaryYAxis = {
        minimum: 0, maximum: 300, interval: 50
    };
    this.radius={ bottomLeft: 15, bottomRight: 15, topLeft: 15, topRight:
15 };
    this.pointColorMapping = 'color';
    this.title = 'Tiger Population - 2016';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Stacking total in Angular Chart component

By using the `annotation`, you can show any element in desired view.

To show the total value in data points, follow the given steps:

Step 1:

Change the element value in chart by using the `annotationRender` event. In this event, you can get the stacked value of the series and change the element value to show the total value of the stacking series.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { GridModule } from '@syncfusion/ej2-angular-grids'
import { PageService } from '@syncfusion/ej2-angular-grids'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { PieSeriesService, AccumulationTooltipService,
AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {
    LineSeriesService, DateTimeService, DataLabelService,
    StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ScrollBarService,
    ChartAnnotationService, LegendService, TooltipService, StripLineService,
    SelectionService, ScatterSeriesService, ZoomService,
    ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { IAnnotationRenderEventArgs, Series } from '@syncfusion/ej2-angular-
charts';
import { ChartComponent } from '@syncfusion/ej2-angular-charts';
@Component({
    imports: [

```

```

        ChartModule, ChartAllModule, AccumulationChartAllModule,
        AccumulationChartModule, GridModule, DialogModule
    ],
    providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
        DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
        StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
        LegendService, TooltipService, StripLineService,
        PieSeriesService, AccumulationTooltipService, ScrollBarService,
        AccumulationDataLabelService, SelectionService, ScatterSeriesService,
        PageService, AreaSeriesService, RangeAreaSeriesService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart #chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'
(annotationRender) = 'annotationRender($event)'>
        <e-annotations>
            <e-annotation content='<div id="point1" style="font-
size:11px;font-weight:bold;color:gray;fill:gray;"><span>12</span></div>'
                x='Jamesh' y='11' coordinateUnits='Point' region='Series'>
            </e-annotation>
            <e-annotation content='<div id="point1" style="font-
size:11px;font-weight:bold;color:gray;fill:gray;"><span>12</span></div>'
                x='Michael' y='10' coordinateUnits='Point' region='Series'>
            </e-annotation>
            <e-annotation content='<div id="point1" style="font-
size:11px;font-weight:bold;color:gray;fill:gray;"><span>12</span></div>'
                x='John' y='12' coordinateUnits='Point' region='Series'>
            </e-annotation>
        </e-annotations>
        <e-series-collection>
            <e-series [dataSource]='data1' type='StackingColumn' xName='x'
yName='y' name='Apple' [marker]='marker'></e-series>
            <e-series [dataSource]='data2' type='StackingColumn'
xName='x' yName='y' name='Orange' [marker]='marker'></e-series>
            <e-series [dataSource]='data3' type='StackingColumn'
xName='x' yName='y' name='Grapes' [marker]='marker'></e-series>
        </e-series-collection>
    </ejs-chart>`
    })
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public data1?: Object[];
    public data2?: Object[];
    public data3?: Object[];
    public title?: string;
    public marker?: Object;
    public primaryYAxis?: Object;
    public i:number = 0;
    @ViewChild('chart')
    public chart?:ChartComponent;
    public annotationRender(args: IAnnotationRenderEventArgs):void{
        let length = this.chart!.series.length - 1;
        let value = ((this.chart as ChartComponent ).visibleSeries[length
as any).stackedValues.endValues[this.i];
        this.i += (this.i == length) ? -length : 1;
        args.content.children[0].children[0].innerHTML = value.toString();
    }
}

```

```

ngOnInit(): void {
    this.data1 = [{ x: 'Jamesh', y: 5 }, { x: 'Michael', y: 4 }, { x:
'John', y: 5 }];
    this.data2 = [{ x: 'Jamesh', y: 4 }, { x: 'Michael', y: 3 }, { x:
'John', y: 4 }];
    this.data3 = [{ x: 'Jamesh', y: 1 }, { x: 'Michael', y: 2 }, { x:
'John', y: 2 }];
    this.primaryXAxis = {
        title: 'Manager',
        valueType: 'Category', interval: 1, majorGridLines: { width: 0 }
    };
    this.marker = { dataLabel: { visible: true, position: 'Top', font: {
fontWeight: '600', color: '#ffffff' }}};
    this.title = 'Fruit Consumption';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Selected data grid in Angular Chart component

By using the [dragComplete](#), you can get the selected data values for range selection.

To display the selected data value, follow the given steps:

Step 1:

Get the selected data point values and display the values through grid component by using the [dragComplete](#) event.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { GridModule } from '@syncfusion/ej2-angular-grids'
import { PageService } from '@syncfusion/ej2-angular-grids'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { PieSeriesService, AccumulationTooltipService,
AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {
    LineSeriesService, DateTimeService, DataLabelService,
    StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ScrollBarService,
    ChartAnnotationService, LegendService, TooltipService, StripLineService,
    SelectionService, ScatterSeriesService, ZoomService,
    ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { IDragCompleteEventArgs } from '@syncfusion/ej2-angular-charts';

```

```

import { GridComponent } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [
    ChartModule, ChartAllModule, AccumulationChartAllModule,
    AccumulationChartModule, GridModule, DialogModule
  ],
  providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
    DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
    LegendService, TooltipService, StripLineService,
    PieSeriesService, AccumulationTooltipService, ScrollBarService,
    AccumulationDataLabelService, SelectionService, ScatterSeriesService,
    PageService, AreaSeriesService, RangeAreaSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[selectionMode]='selectionMode' (dragComplete)='dragComplete($event)'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Scatter' xName='x'
yName='y' name='Product A' [marker]='marker'></e-series>>
  </e-series-collection>
</ejs-chart>
<ejs-grid #grid>
  <e-columns>
    <e-column field='x' headerText='X' textAlign='right'
type='string'></e-column>
    <e-column field='y' headerText='Y'
type='number'></e-column>
  </e-columns>
</ejs-grid>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  public primaryYAxis?: Object;
  public selectionMode?: Object;
  @ViewChild('grid')
  public grid?: GridComponent;
  public dragComplete(args: IDragCompleteEventArgs):void {
    this.grid!.dataSource = args.selectedDataValues[0];
    this.grid!.refresh();
  }
  ngOnInit(): void {
    this.chartData =[
      { x: 1971, y: 50 }, { x: 1972, y: 20 }, { x: 1973, y: 63 }, { x:
1974, y: 81 }, { x: 1975, y: 64 },
      { x: 1976, y: 36 }, { x: 1977, y: 22 }, { x: 1978, y: 78 }, { x:
1979, y: 60 }, { x: 1980, y: 41 },
      { x: 1981, y: 62 }, { x: 1982, y: 56 }, { x: 1983, y: 96 }, { x:
1984, y: 48 }, { x: 1985, y: 23 },
      { x: 1986, y: 54 }, { x: 1987, y: 73 }, { x: 1988, y: 56 }, { x:
1989, y: 67 }, { x: 1990, y: 79 },
      { x: 1991, y: 18 }, { x: 1992, y: 78 }, { x: 1993, y: 92 }, { x:
1994, y: 43 }, { x: 1995, y: 29 },

```



```

    { x: 1996, y: 14 }, { x: 1997, y: 85 }, { x: 1998, y: 24 }, { x:
1999, y: 61 }, { x: 2000, y: 80 },
    { x: 2001, y: 14 }, { x: 2002, y: 34 }, { x: 2003, y: 81 }, { x:
2004, y: 70 }, { x: 2005, y: 21 },
    { x: 2006, y: 70 }, { x: 2007, y: 32 }, { x: 2008, y: 43 }, { x:
2009, y: 21 }, { x: 2010, y: 63 },
    { x: 2011, y: 9 }, { x: 2012, y: 51 }, { x: 2013, y: 25 }, { x:
2014, y: 96 }, { x: 2015, y: 32 }
    ];
    this.primaryXAxis={
        minimum: 1970,
        maximum: 2016
    };
    this.primaryYAxis = {
        title: 'Sales',
        labelFormat: '{value}%',
        interval: 25,
        minimum: 0,
        maximum: 100,
    };
    this.marker = {
        shape: 'Triangle',
        width: 10,
        height: 10
    };
    this.selectionMode = 'DragXY';
    this.title = 'Profit Comparision of A and B';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Marker customization in Angular Chart component

By using the [pointRender](#), you can customize the marker shape.

To Customize the marker shape, follow the given steps:

Step 1:

Customize the marker shape in each data point by using the [pointRender](#) event.

Using this event, you can set the **shape** value to the argument.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { GridModule } from '@syncfusion/ej2-angular-grids'
import { PageService } from '@syncfusion/ej2-angular-grids'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'

```

```

import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { PieSeriesService, AccumulationTooltipService,
AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {
    LineSeriesService, DateTimeService, DataLabelService,
    StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ScrollBarService,
    ChartAnnotationService, LegendService, TooltipService, StripLineService,
    SelectionService, ScatterSeriesService, ZoomService,
    ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { IPointRenderEventArgs, ChartShape } from '@syncfusion/ej2-angular-charts';
@Component({
    imports: [
        ChartModule, ChartAllModule, AccumulationChartAllModule,
        AccumulationChartModule, GridModule, DialogModule
    ],
    providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
        DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
        StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
        LegendService, TooltipService, StripLineService,
        PieSeriesService, AccumulationTooltipService, ScrollBarService,
        AccumulationDataLabelService, SelectionService, ScatterSeriesService,
        PageService, AreaSeriesService, RangeAreaSeriesService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
(pointRender)=(pointRender($event))>
        <e-series-collection>
            <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='India' width=2 [marker]='marker'></e-series>
        </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public marker?: Object;
    public primaryYAxis?: Object;
    public shapes: string[] = [
        'Diamond', 'Circle', 'Rectangle', 'Line', 'Triangle', 'Rectangle'
    ];
    public pointRender(args: IPointRenderEventArgs): void {
        args.shape = <ChartShape>this.shapes[args.point.index];
    };
    ngOnInit(): void {
        this.chartData = [
            { x: new Date(2005, 0, 1), y: 21 }, { x: new Date(2006, 0, 1),
y: 24 },
            { x: new Date(2007, 0, 1), y: 30 }, { x: new Date(2008, 0, 1),
y: 38 },
            { x: new Date(2009, 0, 1), y: 54 }, { x: new Date(2010,
0, 1), y: 57 },

```

```

];
this.primaryXAxis = {
  title: 'Year',
  valueType: 'DateTime'
};
this.primaryYAxis = {
  title: 'Efficiency',
};
this.marker = { visible: true, width: 10, height: 10, dataLabel: {
visible: true}};
this.title = 'Inflation - Consumer Price';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Legend customization in Angular Chart component

By using the [legendRender](#), you can customize the legend shape.

To Customize the legend shape, follow the given steps:

Step 1:

Set the shape value `args.shape` to the argument to change the legend shape by using the [legendRender](#) event.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from '@syncfusion/ej2-angular-charts'
import { GridModule } from '@syncfusion/ej2-angular-grids'
import { PageService } from '@syncfusion/ej2-angular-grids'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { PieSeriesService, AccumulationTooltipService, AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {
  LineSeriesService, DateTimeService, DataLabelService,
  StackingColumnSeriesService, CategoryService,
  StepAreaSeriesService, SplineSeriesService, ScrollBarService,
  ChartAnnotationService, LegendService, TooltipService, StripLineService,
  SelectionService, ScatterSeriesService, ZoomService,
  ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { ILegendRenderEventArgs } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [

```

```

        ChartModule, ChartAllModule, AccumulationChartAllModule,
        AccumulationChartModule, GridModule, DialogModule
    ],
    providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
        DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
        StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
        LegendService, TooltipService, StripLineService,
        PieSeriesService, AccumulationTooltipService, ScrollBarService,
        AccumulationDataLabelService, SelectionService, ScatterSeriesService,
        PageService, AreaSeriesService, RangeAreaSeriesService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'
(legendRender)='legendRender($event)'>
        <e-series-collection>
            <e-series [dataSource]='chartData' type='StepArea' xName='x'
yName='y' name='Renewable' width=2 [marker]='marker'></e-series>
            <e-series [dataSource]='chartData1' type='StepArea' xName='x'
yName='y' name='Non-Renewable' width=2 [marker]='marker'></e-series>
        </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public chartData1?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    marker: any;
    public legendRender(args: ILegendRenderEventArgs): void {
        if (args.text === 'Renewable') {
            args.shape = 'Circle';
        } else if (args.text === 'Non-Renewable') {
            args.shape = 'Triangle';
        }
    };
    ngOnInit(): void {
        this.chartData = [{ x: 2000, y: 416 }, { x: 2001, y: 490 }, { x:
2002, y: 470 }, { x: 2003, y: 500 },
            { x: 2004, y: 449 }, { x: 2005, y: 470 }, { x: 2006, y: 437
}, { x: 2007, y: 458 }];
        this.chartData1 = [{ x: 2000, y: 180 }, { x: 2001, y: 240 }, { x:
2002, y: 370 }, { x: 2003, y: 200 },
            { x: 2004, y: 229 }, { x: 2005, y: 210 }, { x: 2006, y: 337
}, { x: 2007, y: 258 }];
        this.primaryXAxis = {
            title: 'Year',
            valueType: 'Double'
        };
        this.primaryYAxis = {
            title: 'Production (Billion as kWh)',
        };
        this.title = 'Electricity- Production';
    }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Tool tip table in Angular Chart component

You can show the tooltip as table by using template property in tooltip.

Follow the given steps to show the table tooltip,

Step 1:

Initialize the tooltip template div as shown in the following html page,

,

```
<script id="Female-Material" type="text/x-template">
```

```
<div id='templateWrap'>
```

Female	
\$(x):	\$(y)

```
</div>
```

```
</script>
```

,

Step 2:

To show that tooltip template, set the element id to the `template` property in tooltip.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { LineSeriesService, DateTimeService,
  DataLabelService, StackingColumnSeriesService, CategoryService, ChartShape,
  StepAreaSeriesService, ChartAnnotationService, LegendService,
  TooltipService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { IPointRenderEventArgs } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ LineSeriesService, DateTimeService, DataLabelService,
    StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, ChartAnnotationService, LegendService,
    TooltipService ],
  standalone: true,
```

```

        selector: 'app-container',
        template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[tooltip]='tooltip'>
            <e-series-collection>
                <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Female' width=2 [marker]='marker'></e-series>
            </e-series-collection>
        </ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public marker?: Object;
        public tooltip?: Object;
        public primaryYAxis?: Object;
        ngOnInit(): void {
            this.chartData = [
                { x: 2010, y: 990 }, { x: 2011, y: 1010 },
                { x: 2012, y: 1030 }, { x: 2013, y: 1070 },
                { x: 2014, y: 1105 }, { x: 2015, y: 1138 },
                { x: 2016, y: 1155 }
            ];
            this.primaryXAxis = {
                minimum: 2010, maximum: 2016,
                edgeLabelPlacement: 'Shift',
            };
            this.primaryYAxis = {
                minimum: 900, maximum: 1300,
                labelFormat: '{value}M',
            };
            this.marker = { visible: true, width: 10, height: 10, shape:
'Rectangle' };
            this.tooltip = {
                enable: true,
                template: '#Female-Material'
            };
            this.title = 'Population of India ( 2010 - 2016 )';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Footer in Angular Chart component

By using **annotation**, you can place any html elements to chart in a desired view.

To create footer and watermark for chart, follow the given steps:

Step 1:

Initialize the custom elements by using the `annotation` property.

By using the `content` option of the annotation object, you can specify the id of the element that needs to be displayed in the chart.

Use the `content` option of the annotation object to create watermark text for chart. The specified content **syncfusion** needs to be displayed in chart in the specified coordinate unit.

```
`bash
```

```
watermark for chart
```

```
<e-annotations>
```

```
<e-annotation content='<div id="chart_cloud" style="font-size:450%; opacity: 0.3;" >syncfusion</div>'
```

```
x='Wed' y= 20 coordinateUnits= 'Point' horizontalAlignment='Center'>
```

```
</e-annotation>
```

```
</e-annotations>
```

```
,
```

Use the `x` and `y` option of the annotation object to create footer for chart.

```
`bash
```

```
<e-annotations>
```

```
footer for chart
```

```
<e-annotation content='<div id="chart" > <a href="https://www.syncfusion.com" target="_blank">www.syncfusion.com</a></div>'
```

```
x=400 y=440 coordinateUnits='Pixel' horizontalAlignment='Center'>
```

```
</e-annotation>
```

```
</e-annotations>
```

```
,
```

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { PieSeriesService, AccumulationTooltipService,
AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {
  LineSeriesService, DateTimeService, DataLabelService,
  StackingColumnSeriesService, CategoryService,
  StepAreaSeriesService, SplineSeriesService, ScrollBarService,
  ChartAnnotationService, LegendService, TooltipService, StripLineService,
  SelectionService, ScatterSeriesService, ZoomService,
  ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
```

```

imports: [
    ChartModule, ChartAllModule, AccumulationChartAllModule,
    AccumulationChartModule
],
providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
    DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
    LegendService, TooltipService, StripLineService,
    PieSeriesService, AccumulationTooltipService, ScrollBarService,
    AccumulationDataLabelService, SelectionService, ScatterSeriesService,
    AreaSeriesService, RangeAreaSeriesService ],
standalone: true,
selector: 'app-container',
template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis' [title]='title'>
    <e-annotations>
        <e-annotation content='<div id="chart_cloud" style="font-
size:450%; opacity: 0.3;" >syncfusion</div>'
x='Wed' y= 20 coordinateUnits= 'Point'
horizontalAlignment='Center'>
        </e-annotation>
        <e-annotation content='<div id="chart" > <a
href="https://www.syncfusion.com"
target="_blank">www.syncfusion.com</a></div>'
x=400 y=440 coordinateUnits='Pixel'
horizontalAlignment='Center'>
        </e-annotation>
    </e-annotations>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Spline' xName='x'
yName='y' name='Max Temp' width=2 [marker]='marker'></e-series>
    </e-series-collection>
</ejs-chart>`
}))
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public marker?: Object;
    public primaryYAxis?: Object;
    ngOnInit(): void {
        this.chartData = [
            { x: 'Sun', y: 15 }, { x: 'Mon', y: 5 }, { x: 'Tue', y:
32 },
            { x: 'Wed', y: 15 }, { x: 'Thu', y: 29 }, { x: 'Fri', y:
24 },
            { x: 'Sat', y: 18 },
        ];
        this.primaryXAxis = {
            valueType: 'Category',
            interval: 1, majorGridLines: { width: 0 },
            labelIntersectAction: 'Rotate90'
        };
        this.primaryYAxis = {
            minimum: 0,
            maximum: 40,
            interval: 10,

```



```

    };
    this.marker = { visible: true };
    this.title = 'NC Weather Report - 2016';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Threshold in Angular Chart component

You can mark a threshold in chart by using the **stripline**.

By using the start and end properties in **striplines** object, you can mark the threshold line in the horizontal axis of the chart as follows,

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from '@syncfusion/ej2-angular-charts';
import { GridModule } from '@syncfusion/ej2-angular-grids';
import { PageService } from '@syncfusion/ej2-angular-grids';
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts';
import { DialogModule } from '@syncfusion/ej2-angular-popups';
import { PieSeriesService, AccumulationTooltipService, AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts';
import {
  LineSeriesService, DateTimeService, DataLabelService,
  StackingColumnSeriesService, CategoryService,
  StepAreaSeriesService, SplineSeriesService, ScrollBarService,
  ChartAnnotationService, LegendService, TooltipService, StripLineService,
  SelectionService, ScatterSeriesService, ZoomService,
  ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
} from '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule, ChartAllModule, AccumulationChartAllModule,
    AccumulationChartModule, GridModule, DialogModule
  ],
  providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
    DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
    LegendService, TooltipService, StripLineService,
    PieSeriesService, AccumulationTooltipService, ScrollBarService,
    AccumulationDataLabelService, SelectionService, ScatterSeriesService,
    PageService, AreaSeriesService, RangeAreaSeriesService ],
  standalone: true,
  selector: 'app-container',

```

```

    template: `<ejs-chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Runs' [marker]='marker'></e-series>>
    </e-series-collection>
</ejs-chart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public marker?: Object;
    public primaryYAxis?: Object;
    ngOnInit(): void {
      this.chartData = [
        {x: 1, y: 5}, {x: 2, y: 22}, {x: 3, y: 10}, {x: 4, y: 12}, {x:
5, y: 5},
        {x: 6, y: 15}, {x: 7, y: 6}, {x: 8, y: 12}, {x: 9, y: 20}, {x:
10, y: 7}];
      this.primaryXAxis={
        title: 'Overs'
      };
      this.primaryYAxis = {
        title: 'Runs',
        stripLines:[
          { start: 15, end: 15.1, color: '#ff512f', visible: true }
        ]
      };
      this.marker={visible: true}
      this.title = 'India Vs Australia 1st match';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Grid data chart in Angular Chart component

You can visualize the data that returned by grid in chart.

To visualize the data in chart, follow the given steps:

Step 1:

Initialize the grid with datasource.

Step 2:

By using the grid's `actionComplete` event and `getCurrentViewRecords` method, you can get the current page records.

By using the grid's **databound** event, you can update the current page records into the chart's datasource and visualize the grid data in chart.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { GridModule } from '@syncfusion/ej2-angular-grids'
import { PageService, SortService, FilterService, GroupService } from
 '@syncfusion/ej2-angular-grids'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { PieSeriesService, AccumulationTooltipService,
AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import { LineSeriesService, DateTimeService,
DataLabelService, StackingColumnSeriesService, CategoryService, ChartShape,
StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
LegendService, TooltipService, StripLineService,
SelectionService, ScatterSeriesService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { IDragCompleteEventArgs, ChartComponent } from '@syncfusion/ej2-
angular-charts';
import { GridComponent, ActionEventArgs } from '@syncfusion/ej2-angular-
grids';
import { Query, DataManager } from '@syncfusion/ej2-data';
import { orderData } from '../datasource';
@Component({
  imports: [
    ChartModule, AccumulationChartModule, GridModule
  ],
  providers: [ LineSeriesService, DateTimeService, DataLabelService,
StackingColumnSeriesService, CategoryService,
StepAreaSeriesService, SplineSeriesService,
ChartAnnotationService, LegendService, TooltipService, StripLineService,
PieSeriesService, AccumulationTooltipService,
AccumulationDataLabelService, SelectionService, ScatterSeriesService
, PageService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-grid #grid [dataSource]='data' [allowPaging]="true"
[pageSettings]='pageSettings' (dataBound)='dataBound()'
(actionComplete)='actionComplete($event)'>
    <e-columns>
      <e-column field='OrderDate' headerText='Order Date'
width=130 format='yMd' textAlign='right'></e-column>
      <e-column field='Freight' width=120 format='C2'
textAlign='right'></e-column>
    </e-columns>
  </ejs-grid>
  <ejs-chart #chart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-series-collection>
      <e-series type='Line' xName='OrderDate' yName='Freight'
name='dataview' [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
```

```

    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public data?: Object[];
        public title?: string;
        public marker?: Object;
        public primaryYAxis?: Object;
        public pageSettings?: Object;
        @ViewChild('chart')
        public chart?: ChartComponent;
        @ViewChild('grid')
        public grid?: GridComponent;
        ngOnInit(): void {
            this.data = new DataManager(orderData as JSON[]).executeLocal(new
            Query().take(100));
            this.pageSettings = { pageSize: 10 };
        }
        dataBound() {
            this.chart!.primaryXAxis = {
                valueType: 'DateTime',
            };
            this.chart!.series[0].marker = { visible: true };
            this.chart!.series[0].xName = 'OrderDate';
            this.chart!.series[0].yName = 'Freight';
            this.chart!.series[0].dataSource =
            this.grid?.getCurrentViewRecords();
        }
        public actionComplete(args: ActionEventArgs):void {
            if (args.requestType === 'paging') {
                this.chart!.series[0].dataSource =
                this.grid?.getCurrentViewRecords();
                this.chart?.refresh();
            }
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Data label template in Angular Chart component

You can bind text and interior information for a point from dataSource other than x and y value. To change color for the background in the datalabel template, you can use `${point.text}`.

To use point.text, you have to bind the property from dataSource to name in the datalabel options.

Follow the given steps to show the table tooltip,

Step 1:

Initialize the datalabel template div as shown in the following html page,

```

<script id="index" type="text/x-template">
<div id='templateWrap' style="background-color: ${point.text}; border-radius:
3px;"><span>${point.y}</span></div>
</script>

```

Step 2:

To show that datalabel template, set the element id to the `template` property in datalabel.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule } from '@syncfusion/ej2-angular-charts'
import { LineSeriesService, DateTimeService,
DataLabelService, StackingColumnSeriesService, CategoryService,
StepAreaSeriesService, ChartAnnotationService, LegendService,
TooltipService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule
  ],
  providers: [ LineSeriesService, DateTimeService, DataLabelService,
StackingColumnSeriesService, CategoryService,
StepAreaSeriesService, ChartAnnotationService, LegendService,
TooltipService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [title]='title'>
    <e-series-collection>
      <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Female' width=2 [marker]='marker'></e-series>
    </e-series-collection>
  </ejs-chart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  ngOnInit(): void {
    this.chartData = [
      { x: 10, y: 7000, color: 'red' },
      { x: 20, y: 1000, color: 'yellow' },
      { x: 30, y: 12000, color: 'orange' },
      { x: 40, y: 14000, color: 'skyblue' },
      { x: 50, y: 11000, color: 'blue' },
      { x: 60, y: 5000, color: 'green' },
      { x: 70, y: 7300, color: 'pink' },
      { x: 80, y: 9000, color: 'white' },
      { x: 90, y: 12000, color: 'magenta' },
      { x: 100, y: 14000, color: 'purple' },
      { x: 110, y: 11000, color: 'teal' },
    ]
  }
}

```

```

        { x: 120, y: 5000, color: 'gray' },
      ];
      this.marker = { visible: true, dataLabel: { visible: true, name:
'color', template: '#index' } };
      this.title = 'Sales Rate in USA';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Hide tool tip in Angular Chart component

By using the [tooltipRender](#) event, you can cancel the tooltip for unselected series in the chart.

To hide the tooltip value in unselected series, follow the given steps:

Step 1:

By using the [tooltipRender](#) event, you can get the series elements in the arguments. By using this argument we can compare whether seriesElementclasslist is deselected container or not.

If it is true then we cancel the tooltip by setting the value for `args.cancel` as `true`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { GridModule } from '@syncfusion/ej2-angular-grids'
import { PageService } from '@syncfusion/ej2-angular-grids'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { PieSeriesService, AccumulationTooltipService,
AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {
  LineSeriesService, DateTimeService, DataLabelService,
  StackingColumnSeriesService, CategoryService,
  StepAreaSeriesService, SplineSeriesService, ScrollBarService,
  ChartAnnotationService, LegendService, TooltipService, StripLineService,
  SelectionService, ScatterSeriesService, ZoomService,
  ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { ITooltipRenderEventArgs, Series } from '@syncfusion/ej2-angular-
charts';
@Component({
  imports: [
    ChartModule, ChartAllModule, AccumulationChartAllModule,
    AccumulationChartModule, GridModule, DialogModule
  ],

```

```

providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
    LegendService, TooltipService, StripLineService,
    PieSeriesService, AccumulationTooltipService, ScrollBarService,
    AccumulationDataLabelService, SelectionService, ScatterSeriesService,
    PageService, AreaSeriesService, RangeAreaSeriesService ],
standalone: true,
    selector: 'app-container',
    template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis' [title]='title'
    [tooltip]='tooltip' selectionMode='Series'
(tooltipRender)='tooltipRender($event)'>
    <e-series-collection>
        <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Max Temp' width=2 [marker]='marker'></e-series>
        <e-series [dataSource]='chartData1' type='Line' xName='x'
yName='y' name='Max Temp' width=2 [marker]='marker'></e-series>
    </e-series-collection>
    </ejs-chart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public marker?: Object;
    public primaryYAxis?: Object;
    public chartData1: Object[] = [
        { x: new Date(2005, 0, 1), y: 28 }, { x: new Date(2006, 0, 1), y: 44
    },
        { x: new Date(2007, 0, 1), y: 48 }, { x: new Date(2008, 0, 1), y: 50
    },
        { x: new Date(2009, 0, 1), y: 66 }, { x: new Date(2010, 0, 1), y: 78
    },
        { x: new Date(2011, 0, 1), y: 84 }
    ];
    public tooltip: Object = {
        enable: true
    };
    public tooltipRender(args: ITooltipRenderEventArgs): void {
        let series: Series = <Series>(args.series);
        if (series.seriesElement.classList[0] === 'chart-
container_ej2_deselected') {
            args.cancel = true;
        }
    };
    ngOnInit(): void {
        this.chartData =[
            { x: new Date(2005, 0, 1), y: 21 }, { x: new Date(2006, 0, 1), y: 24
        },
            { x: new Date(2007, 0, 1), y: 36 }, { x: new Date(2008, 0, 1), y: 38
        },
            { x: new Date(2009, 0, 1), y: 54 }, { x: new Date(2010, 0, 1), y: 57
        },
            { x: new Date(2011, 0, 1), y: 70 }
        ];
        this.primaryXAxis = {

```

```

        valueType: 'DateTime',
        labelFormat: 'Y',
        intervalType: 'Years',
        edgeLabelPlacement: 'Shift',
        majorGridLines: { width: 0 }
    };
    this.primaryYAxis = {
        labelFormat: '{value}%',
        rangePadding: 'None',
        minimum: 0,
        maximum: 100,
        interval: 20,
        lineStyle: { width: 0 },
        majorTickLines: { width: 0 },
        minorTickLines: { width: 0 }
    };
    this.marker = { visible: true,
        height: 10,
        width: 10 };
    this.title = 'NC Weather Report - 2016';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Dotted line in Angular Chart component

By using **annotation**, you can add dotted lines in the chart.

To add dotted lines in the chart, follow the given steps:

Step 1:

Initialize the custom elements by using the **annotation** property.

By setting **coordinateUnits** value as **point** in annotation object you can placed dotted lines

in the chart based on point x and y values.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { GridModule } from '@syncfusion/ej2-angular-grids'
import { PageService } from '@syncfusion/ej2-angular-grids'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { PieSeriesService, AccumulationTooltipService,
AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {

```



```

    LineSeriesService, DateTimeService, DataLabelService,
    StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ScrollBarService,
    ChartAnnotationService, LegendService, TooltipService, StripLineService,
    SelectionService, ScatterSeriesService, ZoomService,
    ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
  } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule, ChartAllModule, AccumulationChartAllModule,
    AccumulationChartModule, GridModule, DialogModule
  ],
  providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
    DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
    LegendService, TooltipService, StripLineService,
    PieSeriesService, AccumulationTooltipService, ScrollBarService,
    AccumulationDataLabelService, SelectionService, ScatterSeriesService,
    PageService, AreaSeriesService, RangeAreaSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'
[title]='title'
[tooltip]='tooltip'>
  <e-annotations>
    <e-annotation visible = true
      content='<div id = "test" style="border-top:3px dashed
grey;border-top-width: 2px; width: 10000px"></div>'
      x = '2014' y = 50
      coordinateUnits='Point'>
    </e-annotation>
  </e-annotations>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Max Temp'></e-series>
  </e-series-collection>
</ejs-chart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public marker?: Object;
  public primaryYAxis?: Object;
  public tooltip: Object = {
    enable: true
  };
  ngOnInit(): void {
    this.chartData = [
      { x: '2014', y: 34 }, { x: '2015', y: 38 },
      { x: '2016', y: 44 }, { x: '2017', y: 51 },
      { x: '2018', y: 61 }, { x: '2019', y: 76 },
      { x: '2020', y: 82 }
    ];
    this.primaryXAxis = {
      valueType: 'Category'
    }
  }
}

```

```

    };
    this.title = 'NC Weather Report - 2016';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Initial scrollbar in Angular Chart component

By setting `zoomFactor` in `primaryXAxis` and `isZoomed` value as `true` in `load` event and `enableScrollbar` value as `true` in `zoomSettings`, you can make the scrollbar visible in initial rendering of chart.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { GridModule } from '@syncfusion/ej2-angular-grids'
import { PageService } from '@syncfusion/ej2-angular-grids'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { PieSeriesService, AccumulationTooltipService,
 AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {
  LineSeriesService, DateTimeService, DataLabelService,
  StackingColumnSeriesService, CategoryService,
  StepAreaSeriesService, SplineSeriesService, ScrollBarService,
  ChartAnnotationService, LegendService, TooltipService, StripLineService,
  SelectionService, ScatterSeriesService, ZoomService,
  ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { ILoadedEventArgs } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    ChartModule, ChartAllModule, AccumulationChartAllModule,
    AccumulationChartModule, GridModule, DialogModule
  ],
  providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
    DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
    LegendService, TooltipService, StripLineService,
    PieSeriesService, AccumulationTooltipService, ScrollBarService,
    AccumulationDataLabelService, SelectionService, ScatterSeriesService,
    PageService, AreaSeriesService, RangeAreaSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-chart style='display:block' align='center'
id='chartcontainer' [title]='title' [primaryXAxis]='primaryXAxis'
[primaryYAxis]='primaryYAxis'

```

```

        [tooltip]='tooltip'
    (load)='load($event)' [zoomSettings]='zoomSettings' >
        <e-series-collection>
            <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Germany' width=2 [marker]='marker'> </e-series>
        </e-series-collection>
    </ejs-chart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public marker?: Object;
        public primaryYAxis?: Object;
        public tooltip: Object = {
            enable: true
        };
        public load(args: ILoadedEventArgs): void {
            args.chart.zoomModule.isZoomed = true;
        };
        public zoomSettings: Object = {
            mode: 'X',
            enableMouseWheelZooming: true,
            enablePinchZooming: true,
            enableSelectionZooming: true,
            enableScrollbar: true
        };
        ngOnInit(): void {
            this.chartData =[
                { x: new Date(2005, 0, 1), y: 21 },
                { x: new Date(2006, 0, 1), y: 24 },
                { x: new Date(2007, 0, 1), y: 36 },
                { x: new Date(2008, 0, 1), y: 38 },
                { x: new Date(2009, 0, 1), y: 54 },
                { x: new Date(2010, 0, 1), y: 21 },
                { x: new Date(2011, 0, 1), y: 24 },
                { x: new Date(2012, 0, 1), y: 36 },
                { x: new Date(2013, 0, 1), y: 38 },
                { x: new Date(2014, 0, 1), y: 54 },
                { x: new Date(2015, 0, 1), y: 21 },
                { x: new Date(2016, 0, 1), y: 24 },
                { x: new Date(2017, 0, 1), y: 36 },
                { x: new Date(2018, 0, 1), y: 38 },
            ];
            this.primaryXAxis = {
                zoomFactor: 0.3,
                valueType: 'DateTime',
                labelFormat: 'Y',
                intervalType: 'Years',
                edgeLabelPlacement: 'Shift',
            };
            this.primaryYAxis = {
                labelFormat: '{value}%',
                rangePadding: 'None',
                minimum: 0,
                maximum: 100,
                interval: 20,
            };
        }
    }

```

```

    };
    this.marker = { visible: true,
height: 10,
width: 10 };
    this.title = 'NC Weather Report - 2016';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Dialog chart in Angular Chart component

Using the `content` property of the dialog component, you can show the chart in dialog pop-up.

To show the chart in dialog component, follow the given steps:

Step 1:

Initialize the dialog and button components, and then create a basic chart and set the visibility of dialog to `false` when initialize.

By setting the chart `id` in the `content` property of dialog component, you can show chart when clicking the button component.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from '@syncfusion/ej2-angular-charts'
import { GridModule } from '@syncfusion/ej2-angular-grids'
import { PageService } from '@syncfusion/ej2-angular-grids'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { PieSeriesService, AccumulationTooltipService, AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {
  LineSeriesService, DateTimeService, DataLabelService,
  StackingColumnSeriesService, CategoryService,
  StepAreaSeriesService, SplineSeriesService, ScrollBarService,
  ChartAnnotationService, LegendService, TooltipService, StripLineService,
  SelectionService, ScatterSeriesService, ZoomService,
  ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { DialogComponent } from '@syncfusion/ej2-angular-popups';
@Component({
  imports: [
    ChartModule, ChartAllModule, AccumulationChartAllModule,
    AccumulationChartModule, GridModule, DialogModule
  ],

```

```

providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
LegendService, TooltipService, StripLineService,
    PieSeriesService, AccumulationTooltipService, ScrollBarService,
AccumulationDataLabelService, SelectionService, ScatterSeriesService,
    PageService, AreaSeriesService, RangeAreaSeriesService ],
standalone: true,
    selector: 'app-container',
    template: `<div id="target">
    <ejs-chart align='center' id='chartcontainer' [title]='title'
[primaryXAxis]='primaryXAxis'>
        <e-series-collection>
            <e-series [dataSource]='data' type='Column' xName='x' yName='y'
name='Germany' width=2>
            </e-series>
        </e-series-collection>
    </ejs-chart>
</div>
<div id='defaultDialog'>
    <ejs-dialog #Dialog [showCloseIcon]='showCloseIcon' [target]='target'
[width]='width' [height]='height'
    [visible]='visible' allowDragging=true header='chart 2'>
        <ng-template #content>
            <ejs-chart align='center' id='chartcontainer1' [title]='title'
[primaryXAxis]='primaryXAxis' width='350' height='250'>
                <e-series-collection>
                    <e-series [dataSource]='data1' type='Column' xName='x' yName='y'
name='UK' width=2 fill="blue"> </e-series>
                </e-series-collection>
            </ejs-chart>
        </ng-template>
    </ejs-dialog>
</div>
<button class="e-control e-btn" id='dlgbtn' (click)="BtnClick($event)">Open
Dialog</button>
`
})
export class AppComponent {
    public data: Object[] = [
        { x: new Date(2005, 0, 1), y: 21 }, { x: new Date(2006, 0, 1), y: 24 },
        { x: new Date(2007, 0, 1), y: 36 }, { x: new Date(2008, 0, 1), y: 38 },
        { x: new Date(2009, 0, 1), y: 54 }, { x: new Date(2010, 0, 1), y: 57 },
        { x: new Date(2011, 0, 1), y: 70 }
    ];
    public data1: Object[] = [
        { x: new Date(2005, 0, 1), y: 28 }, { x: new Date(2006, 0, 1), y: 44 },
        { x: new Date(2007, 0, 1), y: 48 }
    ];
    @ViewChild('Dialog')
    public Dialog?: DialogComponent;
    public visible: boolean = false;
    public showCloseIcon: Boolean = true;
    public width: string = '500px';
    public height: string = '450px';
    public target: Element = document.getElementById('target') as Element;
    public content: string = '<div id="container2"></div>';

```

```
//Initializing Primary X Axis
public primaryXAxis: Object = {
  valueType: 'DateTime',
  labelFormat: 'y',
  intervalType: 'Years',
  edgeLabelPlacement: 'Shift'
};
public title: string = 'Inflation - Consumer Price';
public BtnClick = (event: any): void => {
  this.Dialog?.show();
};
constructor() {
  //code
};
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Series visible in Angular Chart component

By using the `chartMouseClicked` event, you can show the series based on respective legend click. In this event, you can get the legend target id, using which you can get the current series index. Based on the index, you can set value of `visible` to `true` or `false`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { GridModule } from '@syncfusion/ej2-angular-grids'
import { PageService } from '@syncfusion/ej2-angular-grids'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { PieSeriesService, AccumulationTooltipService,
AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {
  LineSeriesService, DateTimeService, DataLabelService,
  StackingColumnSeriesService, CategoryService,
  StepAreaSeriesService, SplineSeriesService, ScrollBarService,
  ChartAnnotationService, LegendService, TooltipService, StripLineService,
  SelectionService, ScatterSeriesService, ZoomService,
  ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { IMouseEventArgs, ChartComponent } from '@syncfusion/ej2-angular-
charts';
@Component({
  imports: [
    ChartModule, ChartAllModule, AccumulationChartAllModule,
    AccumulationChartModule, GridModule, DialogModule
```

```

    ],
    providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
        DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
        StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
        LegendService, TooltipService, StripLineService,
        PieSeriesService, AccumulationTooltipService, ScrollBarService,
        AccumulationDataLabelService, SelectionService, ScatterSeriesService,
        PageService, AreaSeriesService, RangeAreaSeriesService ],
    standalone: true,
    selector: 'app-container',
    template: `
        <ejs-chart style='display:block;' #chart
        [chartArea]='chartArea' [width]='width' align='center' id='chartcontainer'
        [primaryXAxis]='primaryXAxis'
            [primaryYAxis]='primaryYAxis' [title]='title'
        [tooltip]='tooltip' (chartMouseClicked)='chartMouseClicked($event)'>
            <e-series-collection>
                <e-series [dataSource]='data' type='Column' xName='x'
                yName='y' name='Gold' width=2 fill="red" opacity=0.7> </e-series>
                <e-series [dataSource]='data1' type='Column' xName='x'
                yName='y' name='Silver' width=2 fill="orange" opacity=0.7> </e-series>
                <e-series [dataSource]='data2' type='Column' xName='x'
                yName='y' name='Bronze' width=2 fill="grey" opacity=0.7> </e-series>
            </e-series-collection>
        </ejs-chart>`
    ))
export class AppComponent implements OnInit {
    chartArea: any;
    primaryYAxis: any;
    width: any;
    ngOnInit(): void {
    }
    @ViewChild('chart')
    public chart1?: ChartComponent;
    public previousTarget = null;
    public data: Object[] = [
        { x: 'USA', y: 46 }, { x: 'GBR', y: 27 }, { x: 'CHN', y: 26 }
    ];
    public data1: Object[] = [
        { x: 'USA', y: 37 }, { x: 'GBR', y: 23 }, { x: 'CHN', y: 18 }
    ];
    public data2: Object[] = [
        { x: 'USA', y: 38 }, { x: 'GBR', y: 17 }, { x: 'CHN', y: 26 }
    ];
    public primaryXAxis: Object = {
        valueType: 'Category', interval: 1,
    };
    public title: string = 'Olympic Medal Counts - RIO';
    public tooltip: Object = {
        enable: true
    };
    public chartMouseClicked(args: IMouseEventArgs): void {
        var flag = false;
        if (((args.target).indexOf('chart_legend_text') > -1) ||
            ((args.target).indexOf('chart_legend_shape') > -1) ||
            ((args.target).indexOf('chart_legend_shape_marker_') &&
            !(args.target).indexOf('chart_legend_element'))) {
            var ids = ((args.target).indexOf('chart_legend_text') > -1) ?

```

```

        (args.target).split('chart_legend_text_')[1] :
args.target.split('chart_legend_shape_marker_')[1] ||
args.target.split('chart_legend_shape_')[1];
    for (var i = 0; i < this.chart1!.series.length; i++) {
        this.chart1!.series[i].visible = false;
    }
    if (ids == this.previousTarget) {
        for (var j = 0; j < this.chart1!.series.length; j++)
            this.chart1!.series[j].visible = true;
        this.chart1!.series[ids].visible = false;
        this.previousTarget = null;
        flag = true;
    }
    if (!flag)
        this.previousTarget = ids as any;
}
};
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Dynamic chart in Angular Chart component

By using html button, you can add the chart dynamically when click the button.

To add the chart dynamically through button click, follow the given steps:

Step 1:

Initially create the html button.

Then create chart inside of button `onClick` function. Now click the button charts will render based on click count.

The following code sample demonstrates the output.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { PieSeriesService, AccumulationTooltipService,
AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {
    LineSeriesService, DateTimeService, DataLabelService,
    StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ScrollBarService,
    ChartAnnotationService, LegendService, TooltipService, StripLineService,
    SelectionService, ScatterSeriesService, ZoomService,
    ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
} from '@syncfusion/ej2-angular-charts'

```



```

import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule, ChartAllModule, AccumulationChartAllModule,
    AccumulationChartModule
  ],
  providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
    DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
    LegendService, TooltipService, StripLineService,
    PieSeriesService, AccumulationTooltipService, ScrollBarService,
    AccumulationDataLabelService, SelectionService, ScatterSeriesService,
    AreaSeriesService, RangeAreaSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: ` <div><button ej-button id='print' (click)='Add()'>Add
Chart</button><div *ngFor="let item of items"><ejs-chart [id]='id'
[title]='title'>
  <e-series-collection>
    <e-series [dataSource]='chartData' type='Line' xName='x'
yName='y' name='Germany' [marker]='marker'></e-series>
  </e-series-collection>
</ejs-chart></div></div>`,
})
export class AppComponent implements OnInit {
  public i:number = 0;
  public id:string = 'chart-container';
  public chartData?: Object[];
  public marker?: Object;
  public title?: string;
  public items:any = [];
  ngOnInit(): void {
    this.chartData = [{ x: 1, y: 21 }, { x: 2, y: 24 }, { x: 3, y: 36 },
    { x: 4, y: 38 }, { x: 5, y: 54 }, { x: 6, y: 57 }, { x: 7, y: 70 }],
    this.title = 'Inflation - Consumer Price';
    this.marker = { visible: true };
  }
  Add() {
    this.id = 'chart-container' + this.i;
    var item = {
      "id":this.id,
    }
    this.items.push(item);
    this.i++;
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Database data in Angular Chart component

- Create the chart data table using database. You can find the database table below,

	ID	Date	Product_A	Product_B	Product_C
1	1	2000-06-11	10	40	80
2	2	2002-03-07	40	60	82
3	3	2004-03-06	15	80	119
4	4	2006-03-30	20	40	90
5	5	2008-03-08	5	60	100
6	6	2010-03-08	40	80	119

You can assign data from the data base to the chart. The **series.dataSource** property should be provided with the data from the server, it accepts **JavaScript array of objects**.

```
`bash
```

```
[
  { ID: 1, Date: "2000-06-11", ProductA: 10, ProductB: 40, Product_C: 80 },
  { ID: 2, Date: "2002-03-07", ProductA: 40, ProductB: 60, Product_C: 82 },
  { ID: 3, Date: "2004-03-06", ProductA: 15, ProductB: 80, Product_C: 119 },
  { ID: 4, Date: "2006-03-30", ProductA: 20, ProductB: 40, Product_C: 90 },
  { ID: 5, Date: "2008-03-08", ProductA: 5, ProductB: 60, Product_C: 100 },
  { ID: 6, Date: "2010-03-08", ProductA: 40, ProductB: 80, Product_C: 119 }
],
```

- Using the Angular CLI, we have used to service to get data from database

```
`bash
```

```
ng generate service chart
```

```
,
```

The command generates ChartService class in src/app/chart.service.ts as follows:

```
`typescript
```

```
import { Injectable } from '@angular/core';
@Injectable({
  providedIn: 'root',
})
export class ChartService {
  constructor() {}
}
```

- Create a function call inside `ngOnInit()` to fetch the chart data from the service in `app.component.ts` file.

```
`typescript
ngOnInit(): void {
  this.getData();
}

getData(): void {
  this.chartService.get().subscribe(data => {
    this.data = data;
  });
}

`typescript
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

// import the ChartModule for the Chart component
import { ChartModule, DateTimeService, LineSeriesService, DateTimeCategoryService, StripLineService }
from '@syncfusion/ej2-angular-charts';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ChartService } from './chart.service';
import { HttpClientModule, HttpClientJsonpModule } from '@angular/common/http';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule, ChartModule, HttpClientModule, HttpClientJsonpModule,
    AppRoutingModule
  ],
  providers: [ChartService, DateTimeService, LineSeriesService, DateTimeCategoryService,
    StripLineService],
  bootstrap: [AppComponent]
```

```

})
export class AppModule { }
`

Angular HttpClient method returns an Observable.

chart.service.ts
`typescript
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ChartService {
  get(): Observable<any> {
    return this.http.get('YOUR BACKEND URL');
  }
  constructor(private http: HttpClient) { }
  private handleError<T>(operation = 'operation', result?: T) {
    return (error: any): Observable<T> => {
      console.error(error);
      console.log(`${operation} failed: ${error.message}`);
      return (result as any);
    };
  }
}
`

app.component.ts
`typescript
import { Component, OnInit } from '@angular/core';
import { ChartService } from './chart.service';

@Component({
  selector: 'app-root',

```

```

template: `<ejs-chart id="chart-container" [primaryXAxis]='primaryXAxis'[primaryYAxis]='primaryYAxis'
[title]='title'>
<e-series-collection>
<e-series [dataSource]='data' type='Line' xName='Date' [marker]='marker' yName='Product_A'
name='Sales'></e-series>
<e-series [dataSource]='data' type='Line' xName='Date' [marker]='marker' yName='Product_B'
name='Sales'></e-series>
<e-series [dataSource]='data' type='Line' xName='Date' [marker]='marker' yName='Product_C'
name='Sales'></e-series>
</e-series-collection>
</ejs-chart>`,
styleUrls: ['./app.component.sass']
})
export class AppComponent implements OnInit {
public primaryXAxis: object;
public chartData: object[];
public title: string;
public primaryYAxis: object;
public marker: object;
public data: object[];
constructor(private chartService: ChartService) { }
ngOnInit(): void {
this.primaryXAxis = {
valueType: 'DateTime',
title: 'Sales Across Years',
labelFormat: 'yMMM'
};
this.primaryYAxis = {
title: 'Sales Amount in millions(USD)'
};
this.title = 'Average Sales Comparison';
this.marker = { visible: true };
this.getData();
}

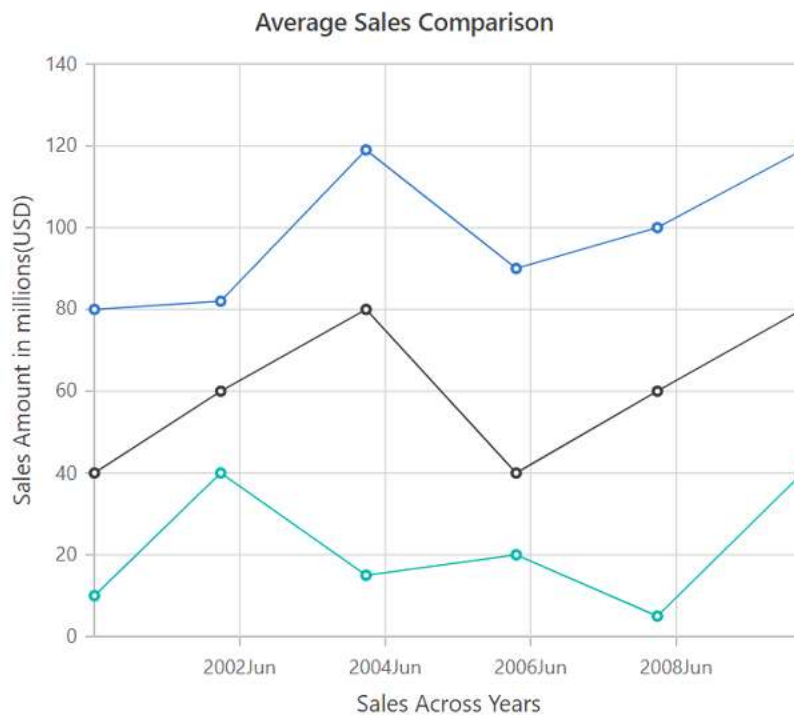
```

```

getData(): void {
  this.chartService.get().subscribe(data => {
    this.data = data;
  });
}

```

The below screenshot shows the chart, that can fetched the data from the server,



Column width in Angular Chart component

By using the [columnWidth](#) and [columnSpacing](#) property in the series of the chart, you can customize the column width and column spacing value for all points of the column series.

To customize the column width and spacing in column series of the chart, follow the given steps:

Step 1:

By setting [columnWidth](#) value between 0 to 1 you can customize the width for every point in the column series.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from '@syncfusion/ej2-angular-charts'
import { GridModule } from '@syncfusion/ej2-angular-grids'
import { PageService } from '@syncfusion/ej2-angular-grids'

```

```

import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { PieSeriesService, AccumulationTooltipService,
AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {
    LineSeriesService, DateTimeService, DataLabelService,
    StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ScrollBarService,
    ChartAnnotationService, LegendService, TooltipService, StripLineService,
    SelectionService, ScatterSeriesService, ZoomService,
    ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
    imports: [
        ChartModule, ChartAllModule, AccumulationChartAllModule,
        AccumulationChartModule, GridModule, DialogModule
    ],
    providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
        DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
        StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
        LegendService, TooltipService, StripLineService,
        PieSeriesService, AccumulationTooltipService, ScrollBarService,
        AccumulationDataLabelService, SelectionService, ScatterSeriesService,
        PageService, AreaSeriesService, RangeAreaSeriesService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart style="display:block;" [chartArea]="chartArea"
[width]="width" align="center"
    id="chartcontainer" [primaryXAxis]="primaryXAxis"
[primaryYAxis]="primaryYAxis"
    [title]="title" [tooltip]="tooltip" >
    <e-series-collection>
    <e-series [dataSource]="data" type="Column" xName="x" yName="y"
name="Gold"
        width="2" columnWidth="0.4">
    </e-series>
    </e-series-collection>
    </ejs-chart>`
})
export class AppComponent {
    public data: Object[] = [
        { x: "USA", y: 4 },
        { x: "GBR", y: 5 },
        { x: "CHN", y: 6 }
    ];
    //Initializing Primary X Axis
    public primaryXAxis: Object = {
        valueType: "Category",
        interval: 1,
        majorGridLines: { width: 0 }
    };
    //Initializing Primary Y Axis
    public primaryYAxis: Object = {
        majorGridLines: { width: 0 },
        majorTickLines: { width: 0 },
        lineStyle: { width: 0 }
    };

```

```

};
public marker: Object = {
  dataLabel: {
    visible: true,
    position: "Top",
    font: { fontWeight: "600", color: "#ffffff" }
  }
};
public title: string = "Olympic Medal Counts - RIO";
public tooltip: Object = {
  enable: true
};
// custom code start
// custom code end
public chartArea: Object = {
  border: {
    width: 0
  }
};
public width: string = "60%";
constructor() {
  //code
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customize scatter chart in Angular Chart component

By using the **shape** property in the marker, you can customize the shape of the scatter series points like Circle, Rectangle, Triangle, Diamond, Cross, HorizontalLine, VerticalLine, Pentagon, InvertedTriangle and Image.

You can customize the width and height of the shapes by using **width** and **height** properties of the marker.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { GridModule } from '@syncfusion/ej2-angular-grids'
import { PageService } from '@syncfusion/ej2-angular-grids'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { PieSeriesService, AccumulationTooltipService,
AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {
  LineSeriesService, DateTimeService, DataLabelService,
  StackingColumnSeriesService, CategoryService,

```



```

        StepAreaSeriesService, SplineSeriesService, ScrollBarService,
        ChartAnnotationService, LegendService, TooltipService, StripLineService,
        SelectionService, ScatterSeriesService, ZoomService,
        ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
    } from '@syncfusion/ej2-angular-charts'
    import { Component, OnInit } from '@angular/core';
    @Component({
        imports: [
            ChartModule, ChartAllModule, AccumulationChartAllModule,
            AccumulationChartModule, GridModule, DialogModule
        ],
        providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
            DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
            StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
            LegendService, TooltipService, StripLineService,
            PieSeriesService, AccumulationTooltipService, ScrollBarService,
            AccumulationDataLabelService, SelectionService, ScatterSeriesService,
            PageService, AreaSeriesService, RangeAreaSeriesService ],
        standalone: true,
        selector: 'app-container',
        template: `
            <ejs-chart style='display:block;' [chartArea]='chartArea'
            align='center' id='chartcontainer' [title]='title'
                [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'>
                <e-series-collection>
                    <e-series [dataSource]='series1' type='Scatter' xName='x'
                    yName='y' opacity='0.6' name='Male' width=2 [marker]='marker1'>
                    </e-series>
                    <e-series [dataSource]='series2' type='Scatter' xName='x'
                    yName='y' opacity='0.6' name='Female' width=2 [marker]='marker2'>
                    </e-series>
                </e-series-collection>
            </ejs-chart>
        `
    })
    export class AppComponent {
        public chartArea: Object = {
            border: {
                width: 0
            }
        };
        //Initializing Primary X Axis
        public primaryXAxis: Object = {
            minimum: 100,
            maximum: 220,
            majorGridLines: { width: 0 },
            edgeLabelPlacement: 'Shift',
            title: 'Height in Inches'
        };
        //Initializing Primary Y Axis
        public primaryYAxis: Object = {
            majorTickLines: {
                width: 0
            },
            lineStyle: {
                width: 0
            },
        },
    }

```

```

        minimum: 0,
        maximum: 100,
        interval: 10,
        title: 'Weight in Pounds',
        rangePadding: 'None'
    };
    public marker1: Object = {
        visible: false,
        width: 28,
        height: 20,
        shape: 'Rectangle',
        dataLabel: {visible: true, position: 'Inner' }
    };
    public marker2: Object = {
        visible: false,
        width: 12,
        height: 12,
        shape: 'Diamond'
    };
    public title: string = 'Height vs Weight';
    public series1: Object[] = [
        { 'x': 131, 'y': 32, text: '131%' }, { 'x': 140, 'y': 52, text: '140%' },
        { 'x': 149, 'y': 82, text: '149%' }, { 'x': 115, 'y': 52, text: '115%' },
        { 'x': 134, 'y': 62, text: '134%' }, { 'x': 183, 'y': 12, text: '183%' },
        { 'x': 155, 'y': 32, text: '155%' }, { 'x': 164, 'y': 22, text: '164%' }
    ];
    public series2: Object[] = [
        { 'x': 115, 'y': 67, text: '115%' },
        { 'x': 138, 'y': 87, text: '138%' },
        { 'x': 166, 'y': 37, text: '166%' },
        { 'x': 122, 'y': 27, text: '122%' },
        { 'x': 126, 'y': 47, text: '126%' },
        { 'x': 119, 'y': 18, text: '119%' },
        { 'x': 141, 'y': 88, text: '141%' },
        { 'x': 180, 'y': 78, text: '180%' }
    ];
    constructor() {
        //code
    };
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customizing point color and data label value

You can customize the point color by using `pointRender` event in the chart. In which we have check the condition based on `yValue` to change the fill color of the point.

By default `dataLabel` values shows `y` values of the datasource. You can customize the `dataLabel` value by using `textRender` event in the chart.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { GridModule } from '@syncfusion/ej2-angular-grids'
import { PageService } from '@syncfusion/ej2-angular-grids'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { PieSeriesService, AccumulationTooltipService,
AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {
    LineSeriesService, DateTimeService, DataLabelService,
    StackingColumnSeriesService, CategoryService,
    StepAreaSeriesService, SplineSeriesService, ScrollBarService,
    ChartAnnotationService, LegendService, TooltipService, StripLineService,
    SelectionService, ScatterSeriesService, ZoomService,
    ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
} from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
import { ITextRenderEventArgs, IPointRenderEventArgs } from
 '@syncfusion/ej2-angular-charts';
@Component({
    imports: [
        ChartModule, ChartAllModule, AccumulationChartAllModule,
        AccumulationChartModule, GridModule, DialogModule
    ],
    providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
        DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,
        StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
        LegendService, TooltipService, StripLineService,
        PieSeriesService, AccumulationTooltipService, ScrollBarService,
        AccumulationDataLabelService, SelectionService, ScatterSeriesService,
        PageService, AreaSeriesService, RangeAreaSeriesService ],
    standalone: true,
    selector: 'app-container',
    template: `
        <ejs-chart style='display:block;' [chartArea]='chartArea'
        align='center' id='chartcontainer' [title]='title'
        [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
        (textRender)='textRender($event)'
        (pointRender)='pointRender($event)' [tooltip]='tooltip' >
            <e-series-collection>
                <e-series [dataSource]='series1' type='Scatter' xName='x'
                yName='y' opacity='0.6' name='Male' width=2 [marker]='marker1'>
            </e-series>
                <e-series [dataSource]='series2' type='Scatter' xName='x'
                yName='y' opacity='0.6' name='Female' width=2 [marker]='marker2'>
            </e-series>
            </e-series-collection>
        </ejs-chart>
    `
})
export class AppComponent {
    public chartArea: Object = {
        border: {

```

```

        width: 0
    }
};
//Initializing Primary X Axis
public primaryXAxis: Object = {
    minimum: 100,
    maximum: 220,
    majorGridLines: { width: 0 },
    edgeLabelPlacement: 'Shift',
    title: 'Height in Inches'
};
//Initializing Primary Y Axis
public primaryYAxis: Object = {
    majorTickLines: {
        width: 0
    },
    lineStyle: {
        width: 0
    },
    minimum: 0,
    maximum: 100,
    interval: 10,
    title: 'Weight in Pounds',
    rangePadding: 'None'
};
public marker1: Object = {
    visible: false,
    width: 28,
    height: 20,
    shape: 'Rectangle',
    dataLabel: {visible: true, position: 'Inner', name: 'text'}
};
public marker2: Object = {
    visible: false,
    width: 12,
    height: 12,
    shape: 'Diamond'
};
public tooltip: Object = {
    enable: true,
    format: 'Weight: <b>${point.x} lbs</b> <br/> Height: <b>${point.y}</b>'
};
public textRender(args: ITextRenderEventArgs): void {
    args.text = args.point.x + ' ';
};
public pointRender(args: IPointRenderEventArgs | any): void {
    if (args.point.y > 80) {
        args.fill='red'
    } else if(args.point.y < 40) {
        args.fill='green'
    }
};
public title: string = 'Height vs Weight';
public series1: Object[] = [
    { 'x': 131, 'y': 32, text: '131%' }, { 'x': 140, 'y': 52, text: '140%' },
    { 'x': 149, 'y': 82, text: '149%' }, { 'x': 115, 'y': 52, text: '115%' },

```

```

    { 'x': 134, 'y': 62, text: '134%' }, { 'x': 183, 'y': 12, text: '183%' },
    { 'x': 155, 'y': 32, text: '155%' }, { 'x': 164, 'y': 22, text: '164%' }
  ]];

  public series2: Object[] = [
    { 'x': 115, 'y': 67, text: '115%' },
    { 'x': 138, 'y': 87, text: '138%' },
    { 'x': 166, 'y': 37, text: '166%' },
    { 'x': 122, 'y': 27, text: '122%' },
    { 'x': 126, 'y': 47, text: '126%' },
    { 'x': 119, 'y': 18, text: '119%' },
    { 'x': 141, 'y': 88, text: '141%' },
    { 'x': 180, 'y': 78, text: '180%' }
  ];
  constructor() {
    //code
  };
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Overlap area in Angular Chart component

You can add a new range area series to show the overlapped area in different color.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, AccumulationChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { GridModule } from '@syncfusion/ej2-angular-grids'
import { PageService } from '@syncfusion/ej2-angular-grids'
import { AccumulationChartModule } from '@syncfusion/ej2-angular-charts'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { PieSeriesService, AccumulationTooltipService,
AccumulationDataLabelService } from '@syncfusion/ej2-angular-charts'
import {
  LineSeriesService, DateTimeService, DataLabelService,
  StackingColumnSeriesService, CategoryService,
  StepAreaSeriesService, SplineSeriesService, ScrollBarService,
  ChartAnnotationService, LegendService, TooltipService, StripLineService,
  SelectionService, ScatterSeriesService, ZoomService,
  ColumnSeriesService, AreaSeriesService, RangeAreaSeriesService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule, ChartAllModule, AccumulationChartAllModule,
    AccumulationChartModule, GridModule, DialogModule
  ],
  providers: [LineSeriesService, DateTimeService, ColumnSeriesService,
    DataLabelService, ZoomService, StackingColumnSeriesService, CategoryService,

```

```

        StepAreaSeriesService, SplineSeriesService, ChartAnnotationService,
        LegendService, TooltipService, StripLineService,
        PieSeriesService, AccumulationTooltipService, ScrollBarService,
        AccumulationDataLabelService, SelectionService, ScatterSeriesService,
        PageService, AreaSeriesService, RangeAreaSeriesService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-chart style='display:block;' [chartArea]='chartArea'
    align='center' [title]='title' [primaryXAxis]='primaryXAxis'
    [primaryYAxis]='primaryYAxis'>
    <e-series-collection>
        <e-series [dataSource]='data' type='Area' xName='x' yName='y'
[marker]='marker' name='Product A' opacity=0.5 width=2
        border-color='transparent'>
        </e-series>
        <e-series [dataSource]='data1' type='Area' xName='x' yName='y'
[marker]='marker' name='Product B' opacity=0.5
        width=2 border-color='transparent'>
        </e-series>
        <e-series [dataSource]='data2' type='RangeArea' xName='x'
high='high' low='low' [marker]='marker' name='Product C'
        opacity=1 width=2 border-color='transparent'>
        </e-series>
    </e-series-collection>
    </ejs-chart>`
    ))
export class AppComponent implements OnInit {
    public chartArea: Object = {
        border: {
            width: 0
        }
    };
    public data: Object[] = [
        { x: new Date(2000, 0, 1), y: 4 }, { x: new Date(2001, 0, 1), y: 3.0 },
        { x: new Date(2002, 0, 1), y: 3.8 }, { x: new Date(2003, 0, 1), y:
3.4 },
        { x: new Date(2004, 0, 1), y: 3.2 }, { x: new Date(2005, 0, 1), y:
3.9 }
    ];
    public data1: Object[] = [
        { x: new Date(2000, 0, 1), y: 2.6 }, { x: new Date(2001, 0, 1), y:
2.8 },
        { x: new Date(2002, 0, 1), y: 2.6 }, { x: new Date(2003, 0, 1), y: 3
},
        { x: new Date(2004, 0, 1), y: 3.6 }, { x: new Date(2005, 0, 1), y: 3
}
    ];
    public data2: Object[] = [
        { x: new Date(2003, 6, 1), high: 3.3, low: 3.3 },
        { x: new Date(2004, 0, 1), high: 3.6, low: 3.2 },
        { x: new Date(2004, 4, 1), high: 3.4, low: 3.4 }
    ];
    //Initializing Primary X Axis
    public primaryXAxis: Object = {
        valueType: 'DateTime',
        labelFormat: 'y',

```

```

        majorGridLines: { width: 0 },
        intervalType: 'Years',
        edgeLabelPlacement: 'Shift'
    };
    //Initializing Primary Y Axis
    public primaryYAxis: Object = {
        title: 'Revenue in Millions',
        labelFormat: '{value}M',
        lineStyle: { width: 0 },
        majorTickLines: { width: 0 },
        minorTickLines: { width: 0 }
    };
    public marker: Object = {
        visible: false
    };
    //Initializing Chart Title
    public title: string = 'Average Sales Comparison';
    constructor() {
        // code
    }
    ngOnInit(): void {
        throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

CheckBox

Getting started with Angular Check box component

This section explains how to create a simple CheckBox, and demonstrate the basic usage of the CheckBox module in an Angular environment.

Dependencies

The following list of dependencies are required to use the CheckBox module in your application.

`typescript

|-- @syncfusion/ej2-angular-buttons

|-- @syncfusion/ej2-angular-base

|-- @syncfusion/ej2-base

|-- @syncfusion/ej2-buttons

`

Setup Angular environment

You can use [Angular CLI](#) to setup your Angular applications. To install Angular CLI use the following command.

```
npm install -g @angular/cli
```

Create an Angular application

Start a new Angular application using below Angular CLI command.

```
ng new my-app
cd my-app
```

Installing Syncfusion CheckBox package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(`>=20.2.36`) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-buttons](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-buttons --save
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-buttons@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-buttons@ngcc --save
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-buttons:"20.2.38-ngcc"
```


Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding CheckBox module

Import CheckBox module into Angular application(app.module.ts) from the package

`@syncfusion/ej2-angular-buttons`.

```
`typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Imported Syncfusion checkbox module from buttons package.
import { CheckBoxModule } from '@syncfusion/ej2-angular-buttons';
import { AppComponent } from './app.component';
@NgModule({
  imports: [ BrowserModule, CheckBoxModule ], // Registering EJ2 Checkbox Module.
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Adding Syncfusion CheckBox component

Modify the template in `app.component.ts` file to render the CheckBox component.

```
`typescript
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<!-- To Render CheckBox. -->
<ejs-checkbox label="Default"></ejs-checkbox>`
})
export class AppComponent { }
```

Adding CSS reference

Add CheckBox component's styles as given below in `style.css`.

```
`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
```

`

Running the application

Run the application in the browser using the following command:

`

ng serve

`

The below example shows a basic CheckBox component,

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CheckBoxModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
    CheckBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!-- To Render CheckBox. -->
    <ejs-checkbox label="Default"></ejs-checkbox>
  </div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Change the CheckBox state

The Essential JS 2 CheckBox contains 3 different states visually, they are:

- Checked
- Unchecked
- Indeterminate

The CheckBox [checked](#) property is used to handle the checked and unchecked state.

In checked state a tick mark will be added to the visualization of CheckBox.

Indeterminate

CheckBox indeterminate state can be set through [indeterminate](#) property. CheckBox indeterminate state masks the real value of CheckBox visually. Checkbox cannot be changed to indeterminate state through the user interface, this state can be achieved only through the property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CheckBoxModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
    CheckBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <ul>
      <!-- checked state. -->
      <li><ejs-checkbox label="Checked State"
[checked]="true"></ejs-checkbox></li>
      <!-- unchecked state. -->
      <li><ejs-checkbox label="Unchecked State"></ejs-
checkbox></li>
      <!-- indeterminate state. -->
      <li><ejs-checkbox label="Indeterminate State"
[indeterminate]="true"></ejs-checkbox></li>
    </ul>
  </div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Label and size in Angular Check box component

This section explains the different sizes and labels.

Label

The CheckBox caption can be defined using the [label](#) property. This reduces the manual addition of label for CheckBox. You can customize the label position before or after the CheckBox through the [labelPosition](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CheckBoxModule } from '@syncfusion/ej2-angular-buttons'
```

```
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [

    CheckBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <ul>
      <!-- Label position - Left. -->
      <li><ejs-checkbox label="Left Side Label"
labelPosition="Before"></ejs-checkbox></li>
      <!-- Label position - Right. -->
      <li><ejs-checkbox label="Right Side Label"
[checked]="true"></ejs-checkbox></li>
    </ul>
  </div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Size

The different CheckBox size are default and small. To reduce the size of default CheckBox to small, set the [cssClass](#) property to `e-small`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CheckBoxModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [

    CheckBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <ul>
      <!-- Small CheckBox. -->
      <li><ejs-checkbox label="Small" cssClass="e-small"></ejs-
checkbox></li>
      <!-- Default CheckBox. -->
      <li><ejs-checkbox label="Default"></ejs-checkbox></li>
    </ul>
  </div>`
})
export class AppComponent { }
```

```

    </div>`
  })
  export class AppComponent { }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [CheckBox customization](#)

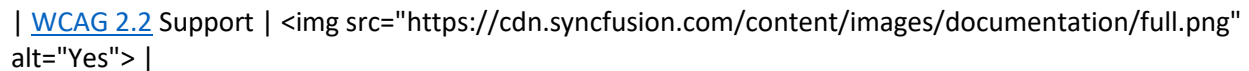
Accessibility in Angular Check box component

The Check box component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Check box component is outlined below.

| Accessibility Criteria | Compatibility |

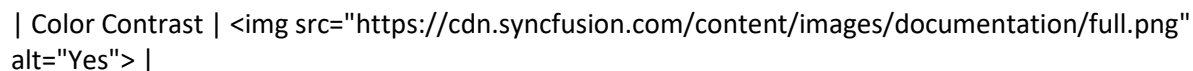
| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes" > |

| [Section 508](#) Support |  alt="Yes" > |

| Screen Reader Support |  alt="Yes" > |

| Right-To-Left Support |  alt="Yes" > |

| Color Contrast |  alt="Yes" > |

| Mobile Device Support |  alt="Yes" > |

| Keyboard Navigation Support |  alt="Yes" > |

| [Accessibility Checker](#) Validation |  alt="Yes" > |

| [Axe-core](#) Accessibility Validation |  alt="Yes" > |

<style>

.post .post-content img {

```
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Check box component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Check box component:

Attributes	Purpose
------------	---------

---	---
-----	-----

aria-disabled	Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable.
---------------	--

Keyboard interaction

The Check box component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Check box component.

Press	To do this
-------	------------

---	---
-----	-----

Space	When the Check box has focus, pressing the Space key changes the state of the CheckBox.
-------	---

Ensuring accessibility

The Check box component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Check box component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the CheckBox component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

How To

Bind data using two way binding in Angular Check box component

Checkbox component supports two way binding.

In this following example, two way binding for CheckBox is illustrated with Switch component. The steps to achieve two way binding in CheckBox are as follows,

- Initialize CheckBox component and bind the checked value using `ngModel` as in the below code using "banana in a box" syntax,

```
`typescript
```

```
<ejs-checkbox #checkbox [(ngModel)]="checked"></ejs-checkbox>
```

```
,
```

- Initialize Switch component and assign the `checked` property value like the below code,

```
`typescript
```

```
<ejs-switch #switch [(checked)]="checked"></ejs-switch>
```

```
,
```

- Now, the changes made in CheckBox will reflect in Switch (i.e When the state of CheckBox is changed to checked state then the Switch state will also change to checked state) and vice versa.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SwitchModule, CheckBoxModule } from '@syncfusion/ej2-angular-buttons'
import { FormsModule } from '@angular/forms'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
    SwitchModule,
    CheckBoxModule,
    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id='container'>
      <table class='tablealign'>
        <thead>
          <th></th>
          <th>
            <h4>Checkbox</h4>
          </th>
          <th>
            <h4>Switch</h4>
          </th>
        </thead>
        <tr>
```

```

        <td class='dataalign'>
            <label>Wi-Fi</label>
        </td>
        <td class='dataalign'>
            <ejs-checkbox #wcheckbox
[ (ngModel) ]="checkedwifi">
                </ejs-checkbox>
            </td>
        <td class='dataalign'>
            <ejs-switch #wswitch
[ (checked) ]="checkedwifi">
                </ejs-switch>
            </td>
        </tr>
        <tr>
            <td class='dataalign'>
                <label>Bluetooth</label>
            </td>
            <td class='dataalign'>
                <ejs-checkbox #bcheckbox
[ (ngModel) ]="checkedbluetooth">
                    </ejs-checkbox>
                </td>
            <td class='dataalign'>
                <ejs-switch #bswitch
[ (checked) ]="checkedbluetooth">
                    </ejs-switch>
                </td>
            </tr>
        </table>
    </div>
</div>`
    })
    export class AppComponent {
        public checkedwifi: boolean = true;
        public checkedbluetooth: boolean = false;
    }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customized checkbox in Angular Check box component

Customize CheckBox Appearance

You can customize the appearance of the CheckBox module using the CSS rules. Define own CSS rules according to your requirement and assign the class name to the [cssClass](#) property.

The background and border color of the CheckBox is customized through the custom classes to create primary, success, warning, danger, and info type of checkbox.

APP.COMPONENT.TS


```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CheckBoxModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [

    CheckBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  // To customize CheckBox appearance
  template: `<div class="e-section-control">
    <ul>
      <!-- Refer the 'e-primary' class details in 'style.css'. -->
      <li><ejs-checkbox label="Primary" cssClass="e-primary"
[checked]="true"></ejs-checkbox></li>
      <!-- Refer the 'e-success' class details in 'style.css'. -->
      <li><ejs-checkbox label="Success" cssClass="e-success"
[checked]="true"></ejs-checkbox></li>
      <!-- Refer the 'e-info' class details in 'style.css'. -->
      <li><ejs-checkbox label="Info" cssClass="e-info"
[checked]="true"></ejs-checkbox></li>
      <!-- Refer the 'e-warning' class details in 'style.css'. -->
      <li><ejs-checkbox label="Warning" cssClass="e-warning"
[checked]="true"></ejs-checkbox></li>
      <!-- Refer the 'e-danger' class details in 'style.css'. -->
      <li><ejs-checkbox label="Danger" cssClass="e-danger"
[checked]="true"></ejs-checkbox></li>
    </ul>
  </div>`
})
export class AppComponent { }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

STYLES.CSS

```

@import 'node_modules/@syncfusion/ej2-base/styles/material.css';
@import 'node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import 'node_modules/@syncfusion/ej2-angular-base/styles/material.css';
@import 'node_modules/@syncfusion/ej2-angular-buttons/styles/material.css';
.e-section-control {
  margin-top: 150px;
}
.e-checkbox-wrapper {
  margin-top: 18px;
}
.e-checkbox-wrapper.e-primary:hover .e-frame.e-check { /* csslint allow:
adjoining-classes */
  background-color: #e03872;
}

```

```

}
.e-checkbox-wrapper.e-success .e-frame.e-check,
.e-checkbox-wrapper.e-success .e-checkbox:focus + .e-frame.e-check { /*
csslint allow: adjoining-classes */
  background-color: #689f38;
}
.e-checkbox-wrapper.e-success:hover .e-frame.e-check { /* csslint allow:
adjoining-classes */
  background-color: #449d44;
}
.e-checkbox-wrapper.e-info .e-frame.e-check,
.e-checkbox-wrapper.e-info .e-checkbox:focus + .e-frame.e-check { /* csslint
allow: adjoining-classes */
  background-color: #2196f3;
}
.e-checkbox-wrapper.e-info:hover .e-frame.e-check { /* csslint allow:
adjoining-classes */
  background-color: #0b7dda;
}
.e-checkbox-wrapper.e-warning .e-frame.e-check,
.e-checkbox-wrapper.e-warning .e-checkbox:focus + .e-frame.e-check { /*
csslint allow: adjoining-classes */
  background-color: #ef6c00;
}
.e-checkbox-wrapper.e-warning:hover .e-frame.e-check { /* csslint allow:
adjoining-classes */
  background-color: #cc5c00;
}
.e-checkbox-wrapper.e-danger .e-frame.e-check,
.e-checkbox-wrapper.e-danger .e-checkbox:focus + .e-frame.e-check { /*
csslint allow: adjoining-classes */
  background-color: #d84315;
}
.e-checkbox-wrapper.e-danger:hover .e-frame.e-check { /* csslint allow:
adjoining-classes */
  background-color: #ba3912;
}
li {
  list-style: none;
  margin: 25px auto;
}
#loader {
  color: #008c8f;
  height: 40px;
  width: 30%;
  position: absolute;
  font-family: 'Helvetica Neue','calibiri';
  font-size:16px;
  top: 45%;
  left: 45%;
}

```

Customize CheckBox frame

CheckBox frame can be customized as per the requirement by adding CSS rules.

In the following example, to-do list is displayed with round checkbox by changing `border-radius` as `100%` by adding `e-custom` class.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CheckBoxModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    CheckBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  // To customize CheckBox appearance
  template: `<div class="e-section-control">
    <ul>
      <li><ejs-checkbox label="Buy Groceries" cssClass="e-
custom" [checked]="true"></ejs-checkbox></li>
      <li><ejs-checkbox label="Pay Rent" cssClass="e-
custom"></ejs-checkbox></li>
      <li><ejs-checkbox label="Make Dinner" cssClass="e-
custom"></ejs-checkbox></li>
      <li><ejs-checkbox label="Finish To-do List Article"
cssClass="e-custom"></ejs-checkbox></li>
    </ul>
  </div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

STYLES.CSS

```
@import 'node_modules/@syncfusion/ej2-base/styles/material.css';
@import 'node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import 'node_modules/@syncfusion/ej2-angular-base/styles/material.css';
@import 'node_modules/@syncfusion/ej2-angular-buttons/styles/material.css';
.e-section-control {
  margin-top: 150px;
}
#container {
  visibility: hidden;
}
#loader {
  color: #008cff;
  height: 40px;
  width: 30%;
  position: absolute;
```

```

    top: 45%;
    left: 45%;
  }
  li {
    list-style: none;
  }
  .e-checkbox-wrapper {
    margin-top: 18px;
  }
  .e-custom .e-frame {
    border-radius: 100%;
  }
}

```

Customize check icon

CheckBox check icon can be customized as per the requirement by adding CSS rules.

In the following example, the check icon can be customized by changing check icon content, background and border color in focus and hovered states by adding `e-checkicon` class.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CheckBoxModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [

    CheckBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  // To customize CheckBox appearance
  template: `<div class="e-section-control">
    <ul>
      <li><ejs-checkbox label="Buy Groceries" cssClass="e-checkicon" [checked]="true"></ejs-checkbox></li>
      <li><ejs-checkbox label="Pay Rent" cssClass="e-checkicon"></ejs-checkbox></li>
      <li><ejs-checkbox label="Make Dinner" cssClass="e-checkicon"></ejs-checkbox></li>
      <li><ejs-checkbox label="Finish To-do List Article" cssClass="e-checkicon"></ejs-checkbox></li>
    </ul>
  </div>`
})
export class AppComponent { }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

STYLES.CSS

```
@import 'node_modules/@syncfusion/ej2-base/styles/material.css';
@import 'node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import 'node_modules/@syncfusion/ej2-angular-base/styles/material.css';
@import 'node_modules/@syncfusion/ej2-angular-buttons/styles/material.css';
.e-section-control {
    margin-top: 150px;
}
#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    top: 45%;
    left: 45%;
}
li {
    list-style: none;
}
@font-face {
    font-family: 'btn-icon';
    src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjltSfgAAAEoAAAAVmNtYXNlH+dZAAABoAAAAEJnbHlm1v4
8pAAAAfgAAQYAGVhZBOPfZcAAADQAAAAANmhoZWEIUQQJAAAArAAAACRobXR4IAAAAAAYAAAAA
gbG9jYQYQNApQAAAHKAAAAEm1heHABFQCqAAABCAAAACBuYW1l071FxAABhAAAAIxcG9zdK9uovo
AAAhEAAAAGaABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAABAAAAAAQAAJ1LUzF8
PPPUACwQAAAAAANG+nFMAAAAAA2D6cUwAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAIAJ4AAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnBgQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAAACAAAAAwAAABQAawABAAAFAAEAC4AAAAEAAQAAQA
A5wb//wAA5wD//wAAAAEABAAAAAEAAgADAAQABQAGAAcAAAAAAAAADgAkADIAhAEuAewCDAAAAAE
AAAAAA2ED9AACAAA3CQGeAsT9PAwB9AH0AAACAAAAAAPHa/QAAwAHAAALIREhASERIQJpAV7+ov3
QAV7+ogwD6PwYA+gAAAEAAAAAA4sD9AACAAATARF0AxgCAP4MA+gAAAAABAAAAAAP0A/QAQwAAExE
fDyE/DxEvDyEPDgWBAgMFBQcICQkLCwMDQ4NAtONDg0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCws
MDA00df0mDQ4NDawLCwkJCACFBQMCA239Jg4NDQ0LCwsJCQgHBQUDAgEBAgMFBQcICQkLCwsNDQ0
OAtO0DQ0NCwsLCQkIBwUFAwIBAQIDBQUHCAkJCwsLDQ0NAAIAAAAAA/MDxQADAIwAADczESMBDwM
VFw8METM3HwQ3Fz8KPQEVBt8LLwg3NT8INS8FNT8NNS8JByU/BDUvCyMPAQytrQH5AgoEAQEBAQg
hERESEyIJCsgQBiEHNQceOZPbDgUICw0LCQUDBAICBAkGAgEBAQMOBAkIBgcDAwEBAQEDAwMJAgE
BAxYLBQQAawMCAgIEBAoBAQEECgcHBgUFBAMDAQEBAQQFBwkFBQUGEf6tDwKEAwIBAQMDCgwVAwc
GDAsNBwdaAYcB3gEFAwN2HwoELDodGxwaLwkIGwz+igEBHwMBAQECAQEEDBgoKDAYICAgFCAkICwU
EBAQFAwYDBwgIDAghCAcGBgYFBQkEAgYCBawJBgUGBwkJCgkICAcLBAIFAwIEBAQFBQcGBwgHBgY
GBgoJCAYCAGeBAQFGMRkaGw0NDA0LIh4xBAQCBAEBAgADAAAAAOKA/MAHABCAJ0AAAEzHwIRDwM
hLwIDNzM/CjUTHwcVIwcVIy8HETcXmz8KNScxBxEfdjsBHQEfdTMhMz8OES8PIz0BLw4hA0EDBQQ
DAQIEBf5eBQQCAW4RDg0LCQgGBQUDBAFEBAMDawIBAQL7Y0EawQCAgIBAYYKChEQDQsJCAcEBAU
CYt8BAQIDBAUFBQcHBwgICQgKjQECAGMEBAUFBgYHBgcIBwGcCAcHBwYGBgUFBAQDAgIBAQEBAgI
DBAQFBQYGBgcHBwgMAQMDAwUFBgYHBwgICQkJ/tQCiwMEBf3XAwYEAgIEBgFoAQEDBQYGBwgIBw0
KhQEiAQEBAgMDAwTV+94BAQECAwMDBAGyAQECBAYHCAgJCgkQCaQC6/47CQkICQcIBwYGBQQEAWI
CUAgHBwcGBgYFBQQAawMBAgIBAwMEBAUFBQcGBwCHCAImCAcHBwYGBgUFBAQDAgIBAdUJCQgICAg
GBwYFBAQDAgEBAAAAAIAAAAAA6cD9AADAaAADchNSElAQcJAScBESNZA078sgGB/uMuAXkBgDb
+1EwMTZcBCD3+ngFiPf7pAxMAAAAAABIA3gABAAAAAEEEEAAAAABAAAAAABAAgAAQABAAAAAA
CAACACQABAAAAAADAAGAEAAABAAAAAEEAGAGAABAAAAAFAAsAIAABAAAAAAGAAGAKwABAAA
AAAAKACwAMwABAAAAAALABIAXwADAAEEECQAAAAIAcQADAAEEECQABABAAcwADAAEEECQACAA4AgwA
```

```

DAAEECQADABAAkQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAEECQAKAFg
AlwADAAEECQALACQBLyBidG4taWNvblJlZ3VsYXJidG4taWNvbmJ0bilpY29uVmVyc2lubiAxLjB
idG4taWNvbkbZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZG1vd3d3LnN
5bmNmdXNpb24uY29tACAAYgB0AG4ALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAAYgB0AG4ALQBpAGM
AbwBuAGIAdABuAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAUADAAYgB0AG4ALQBpAGMABwB
uAEYAbwBuAHQAIAbnAGUAbgBlAHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBuaGMAZgBlAHM
AaQBvAG4AIABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuaGMAZgBlAHMAaQB
vAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgBAGEDAQQBBQE
GAQcBCAEJAAPtZWRpYS1wbGF5C21lZG1hLXBhdXNlDmFycm93aGVhZC1sZWZ0BHN0b3AJbGlrZS0
tLTAXBGNvcHkQLWRvd25sb2FkLTAYLXdmlQAA) format('trueType');
    font-weight: normal;
    font-style: normal;
}
.e-icons {
    font-family: 'btn-icon' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.e-checkbox-wrapper {
    margin-top: 18px;
}
.e-checkicon.e-checkbox-wrapper .e-frame.e-check::before {
    content: '\e703';
}
.e-checkicon.e-checkbox-wrapper .e-check {
    font-size: 8px;
}
.e-checkicon.e-checkbox-wrapper .e-frame.e-check {
    background-color: white;
    border-color: grey;
    color: grey;
}
.e-checkicon.e-checkbox-wrapper:hover .e-frame.e-check {
    background-color: white;
    border-color: grey;
    color: grey;
}
.e-checkicon.e-checkbox-wrapper .e-checkbox:focus + .e-frame.e-check {
    background-color: white;
    border-color: grey;
    box-shadow: none;
    color: grey;
}
.e-checkicon.e-checkbox-wrapper .e-ripple-element {
    background: grey;
}

```

Name and value in form submit in Angular Check box component

The [name](#) attribute of the CheckBox is used to group Checkboxes. When the Checkboxes are grouped in form, the checked items [value](#) attribute will post to the server on form submit which can be retrieved through the name. The [disabled](#) and unchecked CheckBox value will not be sent to the server on form submit.

In the following code snippet, Cricket and Hockey are in the checked state, Tennis is in disabled state and Basketball is in unchecked state. Now, the value that is in checked state only be sent on form submit.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CheckBoxModule, ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
    CheckBoxModule,
    ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  //Name and Value attribute in form submit.
  template: `<div class="e-section-control">
    <form>
      <ul>
        <li><ejs-checkbox name="Sport" value="Cricket"
label="Cricket" [checked]="true"></ejs-checkbox></li>
        <li><ejs-checkbox name="Sport" value="Hockey"
label="Hockey" [checked]="true"></ejs-checkbox></li>
        <li><ejs-checkbox name="Sport" value="Tennis"
label="Tennis" [disabled]="true"></ejs-checkbox></li>
        <li><ejs-checkbox name="Sport" value="Basketball"
label="Basketball"></ejs-checkbox></li>
        <li><button ejs-button
[isPrimary]="true">Submit</button></li>
      </ul>
    </form>
  </div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Right to left in Angular Check box component

CheckBox component has RTL support. This can be achieved by setting `enableRtl` as `true`.

The following example illustrates how to enable right-to-left support in CheckBox component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CheckBoxModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    CheckBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <ejs-checkbox label="Default" [enableRtl]="true"></ejs-checkbox></div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Ej1 api migration in Angular Check box component

This article describes the API migration process of Checkbox component from Essential JS 1 to Essential JS 2.

Properties

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Checkbox label | **Property:** `text`

 <ej-checkbox id="checkbox" text="Checkbox"></ej-checkbox> | **Property:** `label`

 <ejs-checkbox id="checkbox" label="Checkbox"></ejs-checkbox> |

| Checked state | **Property:** `enableTriState` and `checkState`

 <ej-checkbox id="checkbox" [enableTriState]="true" text="Checked state" checkState="check"></ej-checkbox> | **Property:** `checked`

 <ejs-checkbox id="checkbox" [checked]="true" label="Checked state"></ejs-checkbox> |

| Indeterminate state | **Property:** `enableTriState` and `checkState`

 <ej-checkbox id="checkbox" text="Indeterminate state" [enableTriState]="true" checkState="indeterminate"></ej-checkbox> | **Property:** `indeterminate`

 <ejs-checkbox id="indeterminate" [indeterminate]="true" label="Intermediate state"></ejs-checkbox> |

| Adding custom class | **Property:** *cssClass*

 <ej-checkbox id="checkbox" text="Checkbox" cssClass="custom-class"></ej-checkbox> | **Property:** *cssClass*

 <ejs-checkbox id="checkbox" cssClass="custom-class" label="Checkbox"></ejs-checkbox> |

| Label position | Not applicable | **Property:** *labelPosition*

 <ej-checkbox id="checkbox" label="Checkbox" labelPosition="Before"></ej-checkbox> |

| Disabled state | **Property:** *enabled*

 <ej-checkbox id="checkbox" text="Checkbox" [enabled]="false"></ej-checkbox> | **Property:** *disabled*

 <ejs-checkbox id="checkbox" label="Checkbox" disabled="true"></ejs-checkbox> |

| State persistence | **Property:** *enablePersistence*

 <ej-checkbox id="checkbox" text="Checkbox" [enablePersistence]="true"></ej-checkbox> | **Property:** *enablePersistence*

 <ejs-checkbox id="checkbox" label="Checkbox" [enablePersistence]="true"></ejs-checkbox> |

| RTL | **Property:** *enableRTL*

 <ej-checkbox id="checkbox" text="Checkbox" [enableRTL]="true"></ej-checkbox> | **Property:** *enableRtl*

 <ejs-checkbox id="checkbox" label="Checkbox" [enableRtl]="true"></ejs-checkbox> |

| HTML Attributes | **Property:** *htmlAttributes*

 <ej-checkbox id="checkbox" text="Checkbox" [htmlAttributes]="attributes"></ej-checkbox> | Not applicable |

| Id property | **Property:** *id*

 <ej-checkbox id="checkbox" text="sync"></ej-checkbox> | Not applicable |

| Prefix value of Id | **Property:** *idPrefix*

 <ej-checkbox id="checkbox" text="Checkbox" idPrefix="ng"/> | Not applicable |

| Name attribute | **Property:** *name*

 <ej-checkbox id="checkbox" name="conformation"></ej-checkbox> | **Property:** *name*

 <ejs-checkbox id="checkbox" name="conformation"></ejs-checkbox> |

| Value attribute | **Property:** *value*

 <ej-checkbox id="checkbox" name="conformation" value="Received"></ej-checkbox> | **Property:** *value*

 <ejs-checkbox id="checkbox" name="conformation" value="Received"></ejs-checkbox> |

| Show rounded corner | **Property:** *showRoundedCorner*

 <ej-checkbox id="checkbox" text="Checkbox" [showRoundedCorner]="true"></ej-checkbox> | Not applicable |

| Size | **Property:** *size*

 <ej-checkbox id="checkbox" text="Small" size="small"></ej-checkbox> | **Property:** *cssClass*

 <ejs-checkbox id="checkbox" cssClass="e-small" label="Checkbox"></ejs-checkbox> |

| Validation rules | **Property:** *validationRules*

 <ej-checkbox id="checkbox" [validationRules]="rules"></ej-checkbox> | Not applicable |

| Validation message | **Property:** *validationMessage*

 <ej-checkbox id="checkbox" [validationRules]="rules" [validationMessage]="message"></ej-checkbox> | Not applicable |

Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Destroy | **Method:** *destroy*

 <ej-checkbox #checkbox id="checkbox" text="Checkbox"></ej-checkbox>
 @ViewChild('checkbox')
 public checkbox: CheckBoxLayoutComponent;
 this.checkbox.destroy(); | **Method:** *destroy*

 <ejs-checkbox #checkbox id="checkbox" label="Checkbox"></ejs-checkbox>
 @ViewChild('checkbox')
 public checkbox: CheckBoxLayoutComponent;
 this.checkbox.destroy(); |

| Disable the Checkbox | **Method:** *disable*

 <ej-checkbox #checkbox id="checkbox" text="Checkbox"></ej-checkbox>
 @ViewChild('checkbox')
 public checkbox: CheckBoxLayoutComponent;
 this.checkbox.disable(); | Not applicable |

| Enable the Checkbox | **Method:** *enable*

 <ej-checkbox #checkbox id="checkbox" text="Checkbox"></ej-checkbox>
 @ViewChild('checkbox')
 public checkbox: CheckBoxLayoutComponent;
 this.checkbox.enable(); | Not applicable |

| Check state of the Checkbox | **Method:** *isChecked*

 <ej-checkbox #checkbox id="checkbox" text="Checkbox"></ej-checkbox>
 @ViewChild('checkbox')
 public checkbox: CheckBoxLayoutComponent;
 this.checkbox.isChecked(); | Not applicable |

Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| BeforeChange Event | **Events:** *beforeChange*

 ej-checkbox id="checkbox" text="Checkbox" (beforeChange)="beforeChange(\$event)"></ej-checkbox>
 beforeChange(args) {
 / code block */
} | Not applicable |

| Change Event | **Events:** *change*

 <ej-checkbox id="checkbox" text="Checkbox" (change)="change(\$event)"></ej-checkbox>
 change(args) {
 / code block /
} | **Events:** *change*

 <ejs-checkbox id="checkbox" label="Checkbox" (change)="change(\$event)"></ejs-checkbox>
 change(args) {
 / code block /
} |

| created Event | **Events:** *create*

 <ej-checkbox id="checkbox" text="Checkbox" (create)="create(\$event)"></ej-checkbox>
 create(args) {
 / code block /
} | **Events:** *created*

 <ejs-checkbox id="checkbox" label="Checkbox" (created)="created()"></ejs-checkbox>
 function created() {
 / code block /
} |

| Destroy Event | **Events:** *destroy*

 <ej-checkbox id="checkbox" text="Checkbox" (destroy)="destroy(\$event)"></ej-checkbox>
 destroy(args) {
 / code block */
} | Not applicable |

Chips

Getting started with Angular Chips component

This section explains you the steps required to create a simple **Chip** and demonstrate the basic usage of the Chip component in an Angular environment.

Setup Angular Environment

You can use [Angular CLI](#) to setup your Angular applications.

To install Angular CLI use the following command.

```
`bash
npm install -g @angular/cli
`
```

Create an Angular Application

Start a new Angular application using below Angular CLI command.

```
`bash
ng new my-app
cd my-app
`
```

Installing Syncfusion Chips package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(`>=20.2.36`) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-buttons](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-buttons --save
`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-buttons@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-buttons@ngcc --save
`
```

To mention the `ngcc` package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-buttons:"20.2.38-ngcc"
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Registering ChipList Module

Import `ChipList` module into Angular application(`app.module.ts`) from the package `@syncfusion/ej2-angular-buttons` [`src/app/app.module.ts`].

```
`typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// import the ChipListModule for the Chip component
import { ChipListModule } from '@syncfusion/ej2-angular-buttons';
import { AppComponent } from './app.component';

@NgModule({
  //declaration of ej2-angular-buttons module into NgModule
  imports: [ BrowserModule, ChipListModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Adding CSS reference

Add `Chip` component CSS using the following code in [`src/styles.css`].

```
`css
@import "../node_modules/@syncfusion/ej2-angular-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-angular-buttons/styles/material.css";
`
```

Add Chip

Modify the template in [`src/app/app.component.ts`] file to render the `Chip` component. Add the Angular `Chip` by using `<e-chip>` child selector with text attribute inside of `ChipList` component selector `<ejs-chiplist>` in template section of the `app.component.ts` file.

```
`typescript
import { Component, OnInit } from '@angular/core';

@Component({
```

```

selector: 'my-app',
// specifies the template string for the Chip component
template: <ejs-chiplist text="Janet Leverling"></ejs-chiplist>
})
export class AppComponent {
}
`

```

Now, the basic **chip** is included in Angular CLI application.

Run the application

Use the following command to run the application in browser.

```

`javascript
ng serve --open
`

```

The output will appear as follows.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChipListModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
    ChipListModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `<ejs-chiplist text="Janet Leverling"></ejs-chiplist>`
})
export class AppComponent {
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Types in Angular Chips component

The ChipList control has the following types.

- Input Chip
- Choice Chip

- Filter Chip
- Action Chip

Input Chip

Input Chip holds information in compact form. It converts user input into chips.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChipListModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [

    ChipListModule
  ],
  standalone: true,
  selector: 'my-app',
  // specifies the template string for the Chip component
  template: `
    <ejs-chiplist id="chip">
      <e-chips>
        <e-chip text="Andrew"></e-chip>
        <e-chip text="Janet"></e-chip>
        <e-chip text="Laura"></e-chip>
        <e-chip text="Margaret"></e-chip>
      </e-chips>
    </ejs-chiplist>`
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Choice Chip

Choice Chip allows you to select a single chip from the set of ChipList/ChipCollection. It can be enabled by setting the `selection` property to `Single`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChipListModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
```

```

        ChipListModule
    ],
    standalone: true,
    selector: 'my-app',
    // specifies the template string for the Chip component
    template: `
    <ejs-chiplist id="chip" selection="Single">
        <e-chips>
            <e-chip text="Small"></e-chip>
            <e-chip text="Medium"></e-chip>
            <e-chip text="Large"></e-chip>
            <e-chip text="Extra Large"></e-chip>
        </e-chips>
    </ejs-chiplist>`
  })
  export class AppComponent {
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Filter Chip

Filter Chip allows you to select a multiple chip from the set of ChipList/ChipCollection. It can be enabled by setting the `selection` property to `Multiple`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChipListModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [

    ChipListModule
  ],
  standalone: true,
  selector: 'my-app',
  // specifies the template string for the Chip component
  template: `
    <ejs-chiplist id="chip" selection="Multiple">
        <e-chips>
            <e-chip text="Chai"></e-chip>
            <e-chip text="Chang"></e-chip>
            <e-chip text="Aniseed Syrup"></e-chip>
            <e-chip text="Ikura"></e-chip>
        </e-chips>
    </ejs-chiplist>`
  })
  export class AppComponent {
  }

```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Action Chip

The Action Chip triggers the event like click or delete, which helps doing action based on the event.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChipListModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, OnInit } from '@angular/core';
import { ClickEventArgs } from '@syncfusion/ej2-buttons';
@Component({
  imports: [
    ChipListModule
  ],
  standalone: true,
  selector: 'my-app',
  // specifies the template string for the Chip component
  template: `
<ejs-chiplist id="chip" (click)="chipclick($event)">
  <e-chips>
    <e-chip text="Send a text"></e-chip>
    <e-chip text="Set a remainder"></e-chip>
    <e-chip text="Read my emails"></e-chip>
    <e-chip text="Set alarm"></e-chip>
  </e-chips>
</ejs-chiplist>
`
})
export class AppComponent {
  constructor() {}
  chipclick(e: ClickEventArgs) {
    if(e.text){
      alert("you have clicked " + e.text);
    }
  }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```


Deletable Chip

Deletable Chip allows you to delete a chip from ChipList/ChipCollection. It can be enabled by setting the `enableDelete` property to `true`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChipListModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChipListModule
  ],
  standalone: true,
  selector: 'my-app',
  // specifies the template string for the Chip component
  template: `
    <ejs-chiplist id="chip" enableDelete="true">
      <e-chips>
        <e-chip text="Send a text"></e-chip>
        <e-chip text="Set a remainder"></e-chip>
        <e-chip text="Read my emails"></e-chip>
        <e-chip text="Set alarm"></e-chip>
      </e-chips>
    </ejs-chiplist>
  `
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customization in Angular Chips component

This section explains the customization of styles, leading icons, avatar, and trailing icons in Chip control.

Styles

The Chip control has the following predefined styles that can be defined using the `cssClass` property.

Class	Description
e-primary	Represents a primary chip.
e-success	Represents a positive chip.
e-info	Represents an informative chip.

| e-warning | Represents a chip with caution. |

| e-danger | Represents a negative chip. |

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChipListModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [

    ChipListModule
  ],
  standalone: true,
  selector: 'my-app',
  // specifies the template string for the Chip component
  template: `
    <ejs-chiplist id="chip">
      <e-chips>
        <e-chip text="Primary" cssClass="e-primary"></e-chip>
        <e-chip text="Success" cssClass="e-success"></e-chip>
        <e-chip text="Info" cssClass="e-info"></e-chip>
        <e-chip text="Warning" cssClass="e-warning"></e-chip>
        <e-chip text="Danger" cssClass="e-danger"></e-chip>
      </e-chips>
    </ejs-chiplist>
  `
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Leading Icon

You can add and customize the leading icon of chip using the `leadingIconCss` property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChipListModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [

    ChipListModule
  ],
```

```

standalone: true,
selector: 'my-app',
// specifies the template string for the Chip component
template: `
  <ejs-chiplist id="chip">
    <e-chips>
      <e-chip text="Andrew" leadingIconCss='andrew'></e-chip>
      <e-chip text="Janet" leadingIconCss='janet'></e-chip>
      <e-chip text="Laura" leadingIconCss='laura'></e-chip>
      <e-chip text="Margaret" leadingIconCss='margaret'></e-chip>
    </e-chips>
  </ejs-chiplist>
`
})
export class AppComponent {
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Avatar

You can add and customize the avatar of chip using the `avatarIconCss` property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChipListModule } from '@syncfusion/ej2-angular-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChipListModule
  ],
  standalone: true,
  selector: 'my-app',
  // specifies the template string for the Chip component
  template: `
    <ejs-chiplist id="chip">
      <e-chips>
        <e-chip text="Andrew" avatarIconCss='andrew'></e-chip>
        <e-chip text="Janet" avatarIconCss='janet'></e-chip>
        <e-chip text="Laura" avatarIconCss='laura'></e-chip>
        <e-chip text="Margaret" avatarIconCss='margaret'></e-chip>
      </e-chips>
    </ejs-chiplist>
`
})
export class AppComponent {
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Avatar Content

You can add and customize the avatar content of chip using the `avatarText` property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChipListModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChipListModule
  ],
  standalone: true,
  selector: 'my-app',
  // specifies the template string for the Chip component
  template: `
    <ejs-chiplist id="chip">
      <e-chips>
        <e-chip text="Andrew" avatarText= 'A'></e-chip>
        <e-chip text="Janet" avatarText= 'J'></e-chip>
        <e-chip text="Laura" avatarText= 'L'></e-chip>
        <e-chip text="Margaret" avatarText= 'M'></e-chip>
      </e-chips>
    </ejs-chiplist>`
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Trailing Icon

You can add and customize the trailing icon of chip using the `trailingIconCss` property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChipListModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, OnInit } from '@angular/core';
```

```
@Component({
  imports: [

    ChipListModule
  ],
  standalone: true,
  selector: 'my-app',
  // specifies the template string for the Chip component
  template: `
    <ejs-chiplist id="chip">
      <e-chips>
        <e-chip text="Andrew" trailingIconCss= 'e-dlt-btn'></e-chip>
        <e-chip text="Janet" trailingIconCss= 'e-dlt-btn'></e-chip>
        <e-chip text="Laura" trailingIconCss= 'e-dlt-btn'></e-chip>
        <e-chip text="Margaret" trailingIconCss= 'e-dlt-btn'></e-chip>
      </e-chips>
    </ejs-chiplist>`
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Outline Chip

Outline chip has the border with the background transparent. It can be set using the `cssClass` property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChipListModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [

    ChipListModule
  ],
  standalone: true,
  selector: 'my-app',
  // specifies the template string for the Chip component
  template: `
    <ejs-chiplist id="chip" cssClass="e-outline">
      <e-chips>
        <e-chip text="Chai"></e-chip>
        <e-chip text="Chang"></e-chip>
        <e-chip text="Aniseed Syrup"></e-chip>
        <e-chip text="Ikura"></e-chip>
      </e-chips>
    </ejs-chiplist>`
})
```

```

<ejs-chiplist id="chipset" cssClass="e-outline" enableDelete="true">
  <e-chips>
    <e-chip text="Andrew"></e-chip>
    <e-chip text="Janet"></e-chip>
    <e-chip text="Laura"></e-chip>
    <e-chip text="Margaret"></e-chip>
  </e-chips>
</ejs-chiplist>`
  })
  export class AppComponent {
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Style in Angular Chips component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the chip text

Use the following CSS to customize the chip text properties.

```

`css
.e-chip .e-chip-text {
font-size: 20px;
color: black;
font-weight: normal;
}
`

```

Customizing the chip icon

Use the following CSS to customize the chip icon properties.

```

`css
.e-chip .e-icon {
background-image: url('https://ej2.syncfusion.com/demos/src/chips/images/laura.png');
opacity: 0.8;
}
`

```

Customizing the chip delete button

Use the following CSS to customize the chip delete button.

```

`css

```

```
.e-chip-list .e-chip .e-chip-delete.e-dlt-btn {  
  color: #e3165b;  
  font-size: 12px;  
}  
`
```

Customizing the chip outline

Use the following CSS to customize the chip outline.

```
`css  
.e-chip-list .e-chip.e-outline {  
  border-color: #e3165b;  
  border-width: 3px;  
}  
`
```

Customizing the chip on selection

Use the following CSS to customize the chip on selection.

```
`css  
  
/ To customize single chip on selection /  
.e-chip-list.e-selection .e-chip.e-active {  
  background-color: #ffca1c;  
  color: #e3165b;  
}  
  
/ To customize multiple chip on selection /  
.e-chip-list .e-chip.e-active {  
  background-color: #e3165b;  
  color: white;  
}  
`
```

Customizing the chip avatar text

Use the following CSS to customize the chip avatar text properties.

```
`css  
.e-chip-list .e-chip .e-chip-avatar {  
  background-color: #d51a1a;  
  color: #fafafa;  
}  
`
```

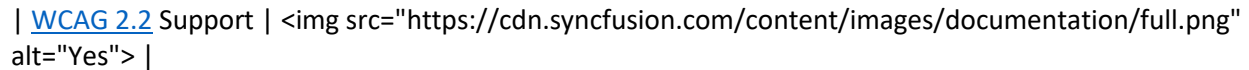
Accessibility in Angular Chips component

The Chips component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Chips component is outlined below.

| Accessibility Criteria | Compatibility |

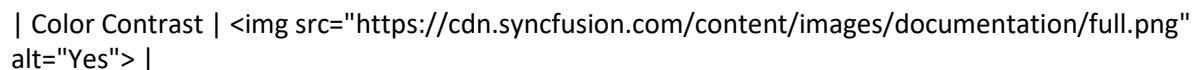
| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes"> |

| [Section 508](#) Support |  alt="Yes"> |

| Screen Reader Support |  alt="Yes"> |

| Right-To-Left Support |  alt="Yes"> |

| Color Contrast |  alt="Yes"> |

| Mobile Device Support |  alt="Yes"> |

| Keyboard Navigation Support |  alt="Yes"> |

| [Accessibility Checker](#) Validation |  alt="Yes"> |

| [Axe-core](#) Accessibility Validation |  alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> alt="Yes"> - All features of the component meet the requirement.</div>

<div> alt="Intermediate"> - Some features of the component do not meet the requirement.</div>

<div> alt="No"> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Chips component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Chips component:

Attributes	Purpose
---	---
<code>role=checkbox</code>	Indicates the ChipList component wrapper element as <code>checkbox</code> .
<code>role=option</code>	Used to convey a significant and contextual message to the user(ChipList).
<code>role=button</code>	Used to convey a significant and contextual message to the user(Single Chip).
<code>aria-label</code>	Provides an accessible name for the Chip.
<code>aria-selected</code>	Indicates the element is selected.
<code>aria-disabled</code>	Indicates element is perceivable but disabled.
<code>aria-multiselectable</code>	Indicates multiple items to be selected.

Keyboard interaction

The Chips component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Chips component.

Keyboard shortcuts	Actions
----- -----	-----
Enter / Space	Selects the targeted chip from the ChipList/ChipCollection.
Delete / Backspace	Deletes the targeted chip from the ChipList/ChipCollection.

Ensuring accessibility

The Chips component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Chips component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Chips component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

3D Circular Chart

Getting started with Angular 3D Circular Chart component

This section explains you the steps required to create a simple `Angular 3D Circular Chart` and demonstrate the basic usage of the 3D Circular Chart component in an Angular environment.

Setup angular environment

You can use [Angular CLI](#) to setup your Angular applications.

To install Angular CLI use the following command.

```
`bash
```

```
npm install -g @angular/cli
```

```
,
```

Create an Angular application

Start a new Angular application using below Angular CLI command.

```
`bash
```

```
ng new my-app
```

```
cd my-app
```

```
,
```

Installing Syncfusion 3D Circular Chart package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages($\geq 20.2.36$) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-charts](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-charts --save
```

```
,
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-charts@ngcc](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-charts@ngcc --save
```

```
,
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
```

```
@syncfusion/ej2-angular-charts:"20.2.38-ngcc"
```

```
,
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Registering 3D Circular Chart module

Import 3D Circular Chart module into Angular application(app.module.ts) from the package `@syncfusion/ej2-angular-charts` [src/app/app.module.ts].

```
`typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// import the CircularChart3DAllModule for the 3D Circular Chart component
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts';
import { AppComponent } from './app.component';
@NgModule({
  //declaration of CircularChart3DAllModule into NgModule
  imports: [ BrowserModule, CircularChart3DAllModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

- Modify the template in `app.component.ts` file to render the `ej2-angular-charts` component

[src/app/app.component.ts].

```
`javascript
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  selector: 'app-container',
  // specifies the template string for the 3D Circular Chart component
  template: <ejs-circularchart3d id='chart-container'></ejs-circularchart3d>,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent { }
```

<!-- markdownlint-disable MD033 -->

Now use the `<code>app-container</code>` in the index.html instead of default one.

```
<app-container></app-container>
```

- Now run the application in the browser using the below command.

```
npm start
```

Pie Series

By default pie series will be rendered on assigning JSON data to the series by using `dataSource` property. Map the field names in the JSON data to the `xName` and `yName` properties of the series.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circularchart3d style='display:block;' align='center'
    tilt='-45' [title]='title' [legendSettings]="legendSettings">
    <e-circularchart3d-series-collection>
    <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
    [dataLabel]='dataLabel'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
    </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public title?: string;
  public legendSettings?: Object;
  public dataLabel?: Object;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Chrome', y: 62.92 },
      { x: 'Internet Explorer', y: 6.12 },
      { x: 'Opera', y: 3.15 },
      { x: 'Edge', y: 5.5 },
      { x: 'Safari', y: 19.97 },
      { x: 'Others', y: 2.34 }];
    this.title = 'Browser Market Shares in November 2023';
    this.legendSettings = { visible: true, position: 'Right' };
    this.dataLabel = {
      visible: true,
      name: 'x',

```

```

        position: 'Outside',
        font: {
            fontWeight: '600'
        },
        connectorStyle: { length: '40px' }
    };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Pie and donut in Angular 3D Circular Chart component

Pie chart

To render a pie series, inject the `PieSeries3DService` module into the `@NgModule.providers`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circularchart3d style='display:block;' align='center'
[tilt]='tilt' [legendSettings]="legendSettings">
  <e-circularchart3d-series-collection>
  <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'>
  </e-circularchart3d-series></e-circularchart3d-series-collection>
  </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public title?: string;
  public legendSettings?: Object;
  public tilt?: number;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 },
      { x: 'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 },
      { x: 'Jul', y: 26 }, { x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }
    ];
    this.legendSettings = { visible: false };
    this.tilt = -45
  }
}

```

```
}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Radius customization

By default, the radius of the pie series will be 80% of the size, which is the minimum of the 3D Circular Chart's width and height. You can customize this by using the `radius` property of the series.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circularchart3d style='display:block;' align='center'
[tilt]='tilt' [legendSettings]="legendSettings">
  <e-circularchart3d-series-collection>
    <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
[radius]='radius'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
  </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public title?: string;
  public legendSettings?: Object;
  public tilt?: number;
  public radius?: string;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 },
      { x: 'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 },
      { x: 'Jul', y: 26 }, { x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 },
      { x: 'Nov', y: 15 }, { x: 'Dec', y: 15 }
    ];
    this.legendSettings = { visible: false };
    this.tilt = -45;
    this.radius = '100%';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Various radius pie chart

You can assign different radii to each slice of the pie by fetching the radius from the data source and using it with the `radius` property in the `series`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circularchart3d style='display:block;' align='center'
    tilt='-15' [title]='title' [legendSettings]="legendSettings"
    [enableAnimation]='enableAnimation'>
    <e-circularchart3d-series-collection>
    <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
    radius='r' innerRadius='20%' [dataLabel]='dataLabel'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
    </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public title?: string;
  public legendSettings?: Object;
  public dataLabel?: Object;
  public enableAnimation?: boolean;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Belgium', y: 551500, r: '110.7' },
      { x: 'Cuba', y: 312685, r: '124.6' },
      { x: 'Dominican Republic', y: 350000, r: '137.5' },
      { x: 'Egypt', y: 301000, r: '150.8' },
      { x: 'Kazakhstan', y: 300000, r: '155.5' },
      { x: 'Somalia', y: 357022, r: '160.6' },
      { x: 'Argentina', y: 505370, r: '100' },
    ];
    this.title = 'Countries compared by population density and total area';
    this.legendSettings = { visible: true };
    this.enableAnimation= true
    this.dataLabel = {
      visible: true,
```

```

        name: 'x',
        position: 'Outside',
        font: {
            fontWeight: '600'
        },
        connectorStyle: { length: '40px' }
    };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Donut chart

To achieve a donut in the pie series, customize the `innerRadius` property of the series. By setting a value greater than 0%, a donut will appear. The `innerRadius` property takes value from 0% to 100% of the pie radius.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
@Component({
    imports: [
        CircularChart3DAllModule
    ],
    providers: [CircularChart3DAllModule],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-circularchart3d style='display:block;' align='center'
[tilt]='tilt' [legendSettings]="legendSettings">
    <e-circularchart3d-series-collection>
    <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
[innerRadius]='innerRadius'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
    </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
    public dataSource?: Object[];
    public title?: string;
    public legendSettings?: Object;
    public innerRadius?: string;
    public tilt?: number;
    ngOnInit(): void {
        this.dataSource = [
            { x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 },
            { x: 'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
            { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 },
            { x: 'Jul', y: 26 }, { x: 'Aug', y: 25 },

```



```

        { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 },
        { x: 'Nov', y: 15 }, { x: 'Dec', y: 15 }]];
    this.innerRadius = '40%';
    this.legendSettings = { visible: false };
    this.tilt = -45
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Text and fill color mapping

The text and the fill color from the data source can be mapped to the 3D Circular Chart using `pointColorMapping` in the series and `name` in the data label, respectively.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circularchart3d style='display:block;' align='center'
[tilt]='tilt' [legendSettings]="legendSettings">
  <e-circularchart3d-series-collection>
    <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
[dataLabel]='dataLabel' [pointColorMapping]='pointColorMapping'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
  </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public dataLabel?: Object;
  public tilt?: number;
  public pointColorMapping?: string
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 3, fill: '#498fff', text: 'January' },
      { x: 'Feb', y: 3.5, fill: '#ffa060', text: 'February' },
      { x: 'Mar', y: 7, fill: '#ff68b6', text: 'March' },
      { x: 'Apr', y: 13.5, fill: '#81e2a1', text: 'April' }
    ];
    this.dataLabel = {
      visible: true,

```

```

    };
    this.pointColorMapping = 'fill'
    this.legendSettings = { visible: false };
    this.tilt = -45
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customization

Individual points in pie chart can be customized using the `pointRender` event.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { CircularChart3DPointRenderEventArgs } from '@syncfusion/ej2-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circularchart3d style='display:block;' align='center'
(pointRender)="pointRender($event)" [tilt]='tilt'
[legendSettings]="legendSettings">
  <e-circularchart3d-series-collection>
  <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'>
  </e-circularchart3d-series></e-circularchart3d-series-collection>
</ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public dataLabel?: Object;
  public tilt?: number;
  public pointRender: Function | any;;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 3 },
      { x: 'Feb', y: 3.5 },
      { x: 'Mar', y: 7 },
      { x: 'Apr', y: 13.5 }];
    this.legendSettings = { visible: false };
    this.pointRender = (args: CircularChart3DPointRenderEventArgs) => {
      if ((args.point.x as string).indexOf('Apr') > -1) {

```

```

        args.fill = '#f4bc42';
    }
    else if ((args.point.x as string).indexOf('Mar') > -1) {
        args.fill = '#DE3D8A';
    }
    else if ((args.point.x as string).indexOf('Feb') > -1) {
        args.fill = '#F7523F';
    }
    else {
        args.fill = '#597cf9';
    }
}
this.tilt= -45
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Data Label in Angular 3D Circular Chart component

A data label refers to a label associated with specific data points. It can be added to a 3D Circular Chart series by enabling the **visible** option in the **dataLabel** property. By default, the labels will arrange themselves smartly to avoid overlapping.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { CircularChart3DTextRenderEventArgs } from '@syncfusion/ej2-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circularchart3d style='display:block;' align='center'
[tilt]='tilt' [legendSettings]="legendSettings">
  <e-circularchart3d-series-collection>
    <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
[dataLabel]='dataLabel'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
  </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public dataLabel?: Object;
  public tilt?: number;

```

```

ngOnInit(): void {
    this.dataSource = [
        { x: 'Jan', y: 13, text: 'Jan: 13' },
        { x: 'Feb', y: 13, text: 'Feb: 13' },
        { x: 'Mar', y: 17, text: 'Mar: 17' },
        { x: 'Apr', y: 13.5, text: 'Apr: 13.5' }];
    this.legendSettings = { visible: false };
    this.dataLabel = {
        visible: true,
    };
    this.tilt = -45
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

To use the data label feature, inject the `CircularChartDataLabel3DService` into the `@NgModule.providers`.

Positioning

Using the `position` property, we can place the data label either `inside` or `outside` the 3D Circular Chart.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { CircularChart3DTextRenderEventArgs } from '@syncfusion/ej2-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circularchart3d style='display:block;' align='center'
[tilt]='tilt' [legendSettings]="legendSettings">
  <e-circularchart3d-series-collection>
    <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
[dataLabel]='dataLabel'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
  </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public dataLabel?: Object;
  public tilt?: number;
  ngOnInit(): void {

```

```

        this.dataSource = [
            { x: 'Jan', y: 13, text: 'Jan: 13' },
            { x: 'Feb', y: 13, text: 'Feb: 13' },
            { x: 'Mar', y: 17, text: 'Mar: 17' },
            { x: 'Apr', y: 13.5, text: 'Apr: 13.5' }];
        this.legendSettings = { visible: false };
        this.dataLabel = {
            visible: true,
            position: 'Outside'
        };
        this.tilt = -45
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Data label template

The label content can be formatted using the template option. Inside the template, placeholder text `${point.x}` and `${point.y}` can be added to display the corresponding data point's x & y value. The data label template can be set using the `template` property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
import { CircularChart3DTextRenderEventArgs } from '@syncfusion/ej2-charts';
@Component({
    imports: [
        CircularChart3DAllModule
    ],
    providers: [CircularChart3DAllModule],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-circularchart3d style='display:block;' align='center'
[tilt]='tilt' [legendSettings]="legendSettings">
    <e-circularchart3d-series-collection>
    <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
[dataLabel]='dataLabel'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
    </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
    public dataSource?: Object[];
    public legendSettings?: Object;
    public dataLabel?: Object;
    public tilt?: number;
    ngOnInit(): void {
        this.dataSource = [

```

```

    { x: 'Jan', y: 13, text: 'Jan: 13' },
    { x: 'Feb', y: 13, text: 'Feb: 13' },
    { x: 'Mar', y: 17, text: 'Mar: 17' },
    { x: 'Apr', y: 13.5, text: 'Apr: 13.5' }]];
    this.legendSettings = { visible: false };
    this.dataLabel = {
      visible: true,
      name: 'text',
      template: "<div id='templateWrap' style='background-
color:#bd18f9;border-radius: 3px;float: right;padding: 2px;line-height:
20px;text-align: center;'>" + "<img
src='https://ej2.syncfusion.com/demos/src/chart/images/sunny.png' />" +
"<div style='color:white; font-family:Roboto; font-style: medium; fontp-
size:14px;float: right;padding: 2px;line-height: 20px;text-align:
center;padding-right:6px'><span>${point.y}</span></div></div>"
    };
    this.tilt = -45
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Connector line

The connector line will be visible when the data label is placed **outside** the chart. It can be customized using properties such as **color**, **width**, **length**, and **dashArray** within the **connectorStyle** property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circularchart3d style='display:block;' align='center'
[tilt]='tilt' [legendSettings]="legendSettings">
  <e-circularchart3d-series-collection>
    <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
[dataLabel]='dataLabel'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
  </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;

```

```

public dataLabel?: Object;
public tilt?: number;
ngOnInit(): void {
    this.dataSource = [
        { x: 'Jan', y: 13, text: 'Jan: 13' },
        { x: 'Feb', y: 13, text: 'Feb: 13' },
        { x: 'Mar', y: 17, text: 'Mar: 17' },
        { x: 'Apr', y: 13.5, text: 'Apr: 13.5' }];
    this.legendSettings = { visible: false };
    this.dataLabel = {
        visible: true,
        name: 'text',
        position: 'Outside',
        connectorStyle: {
            length: '50px',
            width: 2,
            color: '#f4429e',
            dashArray: '5,3'
        }
    };
    this.tilt = -45
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Text mapping

Text from the data source can be mapped using the `name` property within the data label.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
import { CircularChart3DTextRenderEventArgs } from '@syncfusion/ej2-charts';
@Component({
    imports: [
        CircularChart3DAllModule
    ],
    providers: [CircularChart3DAllModule],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-circularchart3d style='display:block;' align='center'
[tilt]='tilt' [legendSettings]="legendSettings">
    <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
[dataLabel]='dataLabel'>
            </e-circularchart3d-series></e-circularchart3d-series-collection>
    </ejs-circularchart3d>`

```

```

    })
    export class AppComponent implements OnInit {
        public dataSource?: Object[];
        public legendSettings?: Object;
        public dataLabel?: Object;
        public tilt?: number;
        ngOnInit(): void {
            this.dataSource = [
                { x: 'Jan', y: 13, text: 'Jan: 13' },
                { x: 'Feb', y: 13, text: 'Feb: 13' },
                { x: 'Mar', y: 17, text: 'Mar: 17' },
                { x: 'Apr', y: 13.5, text: 'Apr: 13.5' }];
            this.legendSettings = { visible: false };
            this.dataLabel = {
                visible: true,
                name: 'text',
            };
            this.tilt = -45
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Format

The data label for the 3D Circular Chart can be formatted using the `format` property. You can utilize global formatting options such as 'n', 'p', and 'c'.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { CircularChart3DTextRenderEventArgs } from '@syncfusion/ej2-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circularchart3d style='display:block;' align='center'
[tilt]='tilt' [legendSettings]="legendSettings">
  <e-circularchart3d-series-collection>
    <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
[dataLabel]='dataLabel'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
  </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {

```



```

public dataSource?: Object[];
public legendSettings?: Object;
public dataLabel?: Object;
public tilt?: number;
ngOnInit(): void {
    this.dataSource = [
        { x: 'Jan', y: 13, text: 'Jan: 13' },
        { x: 'Feb', y: 13, text: 'Feb: 13' },
        { x: 'Mar', y: 17, text: 'Mar: 17' },
        { x: 'Apr', y: 13.5, text: 'Apr: 13.5' }];
    this.legendSettings = { visible: false };
    this.dataLabel = {
        visible: true,
        format: 'n2'
    };
    this.tilt = -45
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Value	Format	Resultant Value	Description
1000	n1	1000.0	The number is rounded to 1 decimal place.
1000	n2	1000.00	The number is rounded to 2 decimal places.
1000	n3	1000.000	The number is rounded to 3 decimal place.
0.01	p1	1.0%	The number is converted to percentage with 1 decimal place.
0.01	p2	1.00%	The number is converted to percentage with 2 decimal place.
0.01	p3	1.000%	The number is converted to percentage with 3 decimal place.
1000	c1	\$1000.0	The currency symbol is appended to number and number is rounded to 1 decimal place.
1000	c2	\$1000.00	The currency symbol is appended to number and number is rounded to 2 decimal place.

Customization

Individual text for the data points in the 3D Circular Chart can be customized using the `textRender` event.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

```

```

import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { CircularChart3DTextRenderEventArgs } from '@syncfusion/ej2-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circularchart3d style='display:block;' align='center'
(textRender)="onTextRender($event)" [tilt]='tilt'
[legendSettings]="legendSettings">
  <e-circularchart3d-series-collection>
    <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
[dataLabel]='dataLabel'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
  </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public dataLabel?: Object;
  public tilt?: number;
  public onTextRender: Function | any;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 13, text: 'Jan: 13' },
      { x: 'Feb', y: 13, text: 'Feb: 13' },
      { x: 'Mar', y: 17, text: 'Mar: 17' },
      { x: 'Apr', y: 13.5, text: 'Apr: 13.5' }];
    this.legendSettings = { visible: false };
    this.dataLabel = {
      visible: true,
      name: 'text',
      position: 'Outside'
    };
    this.onTextRender = (args: CircularChart3DTextRenderEventArgs) => {
      if (args.text.indexOf('Mar') > -1) {
        args.color = 'red';
        args.border.width = 1;
      }
    }
    this.tilt = -45
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Using textRender event

You can customize the data label of a pie chart using the `textRender` event as follows to show the percentage.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { CircularChart3DTextRenderEventArgs } from '@syncfusion/ej2-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circularchart3d style='display:block;' align='center'
(textRender)="onTextRender($event)" [tilt]='tilt'
[legendSettings]="legendSettings">
  <e-circularchart3d-series-collection>
    <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
[dataLabel]='dataLabel'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
  </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public dataLabel?: Object;
  public tilt?: number;
  public onTextRender: Function | any;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 13, text: 'Jan: 13' },
      { x: 'Feb', y: 13, text: 'Feb: 13' },
      { x: 'Mar', y: 17, text: 'Mar: 17' },
      { x: 'Apr', y: 13.5, text: 'Apr: 13.5' }];
    this.legendSettings = { visible: false };
    this.dataLabel = {
      visible: true,
    };
    this.onTextRender = (args: CircularChart3DTextRenderEventArgs) => {
      args.text = args.point.percentage + "%";
    }
    this.tilt = -45
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Using template

You can display the percentage values in the data label of a pie chart using the **template** option.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { CircularChart3DTextRenderEventArgs } from '@syncfusion/ej2-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circularchart3d style='display:block;' align='center'
[tilt]='tilt' [legendSettings]="legendSettings">
  <e-circularchart3d-series-collection>
    <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
[dataLabel]='dataLabel'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
  </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public dataLabel?: Object;
  public tilt?: number;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 13, text: 'Jan: 13' },
      { x: 'Feb', y: 13, text: 'Feb: 13' },
      { x: 'Mar', y: 17, text: 'Mar: 17' },
      { x: 'Apr', y: 13.5, text: 'Apr: 13.5' }];
    this.legendSettings = { visible: false };
    this.dataLabel = {
      visible: true,
      template: "<div
id='dataLabelTemplate'>${point.percentage}%</div>",
    };
    this.tilt= -45
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Empty points in Angular 3D Circular Chart component

Data points containing **null** or **undefined** values are considered empty points. These empty data points are ignored and not plotted in the 3D Circular Chart. You can customize the handling of empty points using the **emptyPointSettings** property in the series. The default mode for empty points is **Gap**. Other supported modes include **Average**, **Drop**, and **Zero**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circularchart3d style='display:block;' align='center'
[tilt]='tilt' [legendSettings]="legendSettings">
  <e-circularchart3d-series-collection>
    <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
[emptyPointSettings]='emptyPointSettings' [dataLabel]='dataLabel'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
  </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public title?: string;
  public legendSettings?: Object;
  public emptyPointSettings?: Object;
  public tilt?: number;
  public dataLabel?: Object;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 },
      { x: 'Mar', y: undefined }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 },
      { x: 'Jul', y: null }, { x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }];
    this.dataLabel = {
      visible: true,
      position: 'Outside'
    };
    this.emptyPointSettings = { mode: 'Zero' };
    this.legendSettings = { visible: false };
    this.tilt = -45
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customization

A specific color for an empty point can be set by using the **fill** property in **emptyPointSettings**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circularchart3d style='display:block;' align='center'
[tilt]='tilt' [legendSettings]="legendSettings">
  <e-circularchart3d-series-collection>
    <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
[emptyPointSettings]='emptyPointSettings'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
  </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public title?: string;
  public legendSettings?: Object;
  public emptyPointSettings?: object;
  public tilt?: number;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 },
      { x: 'Mar', y: undefined }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 },
      { x: 'Jul', y: null }, { x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 }];
    this.emptyPointSettings = { mode: 'Average', fill: 'pink' };
    this.legendSettings = { visible: false };
    this.tilt = -45
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Legend in Angular 3D Circular Chart component

The legend provides information about the data points rendered in the 3D Circular Chart. It can be added by enabling the `visible` option in the `legendSettings` property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    [tilt]='tilt' [legendSettings]="legendSettings">
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'>
        </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public tilt?: number;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 13 },
      { x: 'Feb', y: 13 },
      { x: 'Mar', y: 17 },
      { x: 'Apr', y: 13.5 }
    ];
    this.legendSettings = { visible: true };
    this.tilt = -45;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

To use the legends feature, inject the `CircularChartLegend3DService` into the `@NgModule.providers`.

Position and alignment

By using the `position` property, the legend can be positioned at the `left`, `right`, `top` or `bottom` of the 3D Circular Chart. By default, the legend will be positioned to the right of the 3D Circular Chart.

Additionally, you can align the legend to the `center`, `far` or `near` of the chart using the `alignment` property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    [tilt]='tilt' [legendSettings]="legendSettings">
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
      >
        </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public tilt?: number;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 13 },
      { x: 'Feb', y: 13 },
      { x: 'Mar', y: 17 },
      { x: 'Apr', y: 13.5 }
    ];
    this.legendSettings = { position: 'Top', alignment: 'Near' };
    this.tilt = -45;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```


Legend reverse

You can reverse the order of the legend items by using the `reverse` property in `legendSettings`. By default, the legend for the first series in the collection will be placed first.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    [title]='title' subTitle='subTitle' [tilt]='tilt'
    [legendSettings]="legendSettings">
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
      >
        </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public tilt?: number;
  public title?: string;
  public subTitle?: string;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Saudi Arabia', y: 58 },
      { x: 'Persian Gulf', y: 15 },
      { x: 'Canada', y: 13 },
      { x: 'Venezuela', y: 8 },
      { x: 'Mexico', y: 3 },
      { x: 'Russia', y: 2 },
      { x: 'Miscellaneous', y: 1 }
    ];
    this.legendSettings = {
      visible: true,
      reverse: true
    };
    this.title= 'Oil and other liquid imports in USA',
    this.subTitle = 'In the year 2014 - 2015',
    this.tilt = -45;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Legend shape

To change the legend shape, use the `legendShape` property in the `series`. By default, the legend shape is set to `seriesType`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    [tilt]='tilt' [legendSettings]="legendSettings">
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
        legendShape='Rectangle'>
        </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public tilt?: number;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 13 },
      { x: 'Feb', y: 13 },
      { x: 'Mar', y: 17 },
      { x: 'Apr', y: 13.5 }
    ];
    this.legendSettings = { visible: true };
    this.tilt = -45;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
```

```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Legend size

The legend size can be changed by using the `width` and `height` properties in `legendSettings`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    [tilt]='tilt' [legendSettings]="legendSettings">
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'>
        </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
  })
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public tilt?: number;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 13 },
      { x: 'Feb', y: 13 },
      { x: 'Mar', y: 17 },
      { x: 'Apr', y: 13.5 }
    ];
    this.legendSettings = { width: '150', height: '100', border: {
width: 1, color: 'pink' } };
    this.tilt = -45;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Legend item size

The size of the legend items can be customized by using the `shapeHeight` and `shapeWidth` properties in `legendSettings`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    [tilt]='tilt' [legendSettings]="legendSettings">
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'>
        </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public tilt?: number;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 13 },
      { x: 'Feb', y: 13 },
      { x: 'Mar', y: 17 },
      { x: 'Apr', y: 13.5 }
    ];
    this.legendSettings = {
      shapeHeight: 15,
      shapeWidth: 15
    };
    this.tilt = -45;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Legend paging

Paging will be enabled by default when the legend items exceed the legend bounds. Each legend item can be viewed by navigating between the pages using navigation buttons.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    [tilt]='tilt' [legendSettings]="legendSettings">
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
        >
          </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public tilt?: number;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 },
      { x: 'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 },
      { x: 'Jul', y: 26 }, { x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 },
      { x: 'Nov', y: 15 }, { x: 'Dec', y: 15 }
    ];
    this.legendSettings = { height: '150', width: '80' };
    this.tilt = -45;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Legend text wrap

When the legend text exceeds the container, the text can be wrapped using the `textWrap` property in `legendSettings`. End users can also wrap the legend text based on the `maximumLabelWidth` property in `legendSettings`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circularchart3d style='display:block;' align='center'
[tilt]='tilt' [legendSettings]="legendSettings">
  <e-circularchart3d-series-collection>
    <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
[innerRadius]='innerRadius'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
  </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public title?: string;
  public legendSettings?: Object;
  public innerRadius?: string;
  public tilt?: number;
  ngOnInit(): void {
    this.dataSource = [
      { 'x': 'Net-tution', y: 21 },
      { 'x': 'Private Gifts', y: 8 },
      { 'x': 'All Other', y: 9 },
      { 'x': 'Local Revenue', y: 4 },
      { 'x': 'State Revenue', y: 21 },
      { 'x': 'Federal Revenue', y: 16 },
      { 'x': 'Self-supporting Operations', y: 21 }
    ];
    this.innerRadius = '40%';
    this.legendSettings = {
      visible: true,
      position: 'Right',
      textWrap: 'Wrap',
      maximumLabelWidth: 60
    };
    this.tilt = -45
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
```

```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Legend title

You can set a title for the legend using the `title` property in `legendSettings`. The `size`, `color`, `opacity`, `fontStyle`, `fontWeight`, `fontFamily`, `textAlignment`, and `textOverflow` of the legend title can be customized using the `titleStyle` property in `legendSettings`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    [tilt]='tilt' [legendSettings]="legendSettings">
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'>
        </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public tilt?: number;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 13 },
      { x: 'Feb', y: 13 },
      { x: 'Mar', y: 17 },
      { x: 'Apr', y: 13.5 }
    ];
    this.legendSettings = { visible: true, title: 'Months', position:
    'Bottom' };
    this.tilt = -45;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Arrow page navigation

The page number will always be visible when using legend paging. However, it is now possible to disable the page number and enable page navigation with the left and right arrows. To render the arrow page navigation, set the `enablePages` property to **false**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circularchart3d style='display:block;' align='center'
[tilt]='tilt' [legendSettings]="legendSettings">
  <e-circularchart3d-series-collection>
  <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'>
  </e-circularchart3d-series></e-circularchart3d-series-collection>
  </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public tilt?: number;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 },
      { x: 'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 },
      { x: 'Jul', y: 26 }, { x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 },
      { x: 'Nov', y: 15 }, { x: 'Dec', y: 15 }
    ];
    this.legendSettings = {
      width: '260px',
      height: '50px',
      enablePages: false,
      position: 'Bottom'
    };
    this.tilt = -45;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
```



```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Legend item padding

The `itemPadding` property can be used to adjust the space between the legend items.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    [tilt]='tilt' [legendSettings]="legendSettings">
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'>
        </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
  })
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public tilt?: number;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 },
      { x: 'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 },
      { x: 'Jul', y: 26 }, { x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 },
      { x: 'Nov', y: 15 }, { x: 'Dec', y: 15 }
    ];
    this.legendSettings = {
      width: '260px',
      height: '50px',
      position: 'Bottom',
      itemPadding: 30
    };
    this.tilt = -45;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
```

```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Tooltip in Angular 3D Circular Chart component

The 3D Circular Chart will display details about the points through a tooltip, when the mouse is moved over a specific point. By default, the tooltip is not visible. It can be enabled by using the `enable` property in `tooltip` to `true`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    [tilt]='tilt' [legendSettings]="legendSettings" [tooltip]='tooltip'>
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'>
        </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public title?: string;
  public legendSettings?: Object;
  public tilt?: number;
  public tooltip?: object;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 13 },
      { x: 'Feb', y: 13 },
      { x: 'Mar', y: 17 },
      { x: 'Apr', y: 13.5 }
    ];
    this.legendSettings = { visible: false };
    this.tooltip = { enable: true };
    this.tilt = -45;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
```

```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

To use tooltip feature, inject the `CircularChartTooltip3DService` into the `@NgModule.providers`.

Header

You can specify a header for the tooltip by using the `header` property in `tooltip`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    [tilt]='tilt' [legendSettings]="legendSettings" [tooltip]='tooltip'>
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'>
        </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public title?: string;
  public legendSettings?: Object;
  public tilt?: number;
  public tooltip?: object;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 13 },
      { x: 'Feb', y: 13 },
      { x: 'Mar', y: 17 },
      { x: 'Apr', y: 13.5 }
    ];
    this.legendSettings = { visible: false };
    this.tooltip = { enable: true, header: 'Pie Chart' };
    this.tilt = -45;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Format

By default, the tooltip shows information about the x and y values in points. Additionally, more information can be displayed in the tooltip by using the `format` property. For example, the format `${series.name} : ${point.x}` shows the series name and the point's x value.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    [tilt]='tilt' [legendSettings]="legendSettings" [tooltip]='tooltip'>
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'>
        </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public title?: string;
  public legendSettings?: Object;
  public tilt?: number;
  public tooltip?: object;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 13 },
      { x: 'Feb', y: 13 },
      { x: 'Mar', y: 17 },
      { x: 'Apr', y: 13.5 }
    ];
    this.legendSettings = { visible: false };
    this.tooltip = { enable: true, format: '${point.x} : <b>${point.y}%</b>' }
  }
  this.tilt = -45;
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
```

```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Tooltip template

Any HTML elements can be displayed in the tooltip by using the `template` property in the tooltip.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    [tilt]='tilt' [legendSettings]="legendSettings" [tooltip]='tooltip'>
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'>
        </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
  })
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public title?: string;
  public legendSettings?: Object;
  public tilt?: number;
  public tooltip?: object;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 13 },
      { x: 'Feb', y: 13 },
      { x: 'Mar', y: 17 },
      { x: 'Apr', y: 13.5 }
    ];
    this.legendSettings = { visible: false };
    this.tooltip = {
      enable: true,
      template: "<div id='templateWrap' style='background-color:#bd18f9;border-radius: 3px; float: right;padding: 2px;line-height: 20px;text-align: center;'>"+
        "<img"
        src='https://ej2.syncfusion.com/demos/src/chart/images/sunny.png' />" +
        "<div style='color:white; font-family:Roboto; font-style: medium; font-size:14px;float: right;padding: 2px;line-height: 20px;text-align: center;padding-right:6px'><span>${y}</span></div></div>"
    }
  }
}
```

```

    this.tilt = -45;
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Fixed tooltip

By default, the tooltip tracks the mouse movement, but it can be set to a fixed position using the `location` property in `tooltip`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    [tilt]='tilt' [legendSettings]="legendSettings" [tooltip]='tooltip'>
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'>
        </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
  })
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public title?: string;
  public legendSettings?: Object;
  public tilt?: number;
  public tooltip?: object;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 13 },
      { x: 'Feb', y: 13 },
      { x: 'Mar', y: 17 },
      { x: 'Apr', y: 13.5 }
    ];
    this.legendSettings = { visible: false };
    this.tooltip = { enable: true, location: { x: 200, y: 20 } }
    this.tilt = -45;
  }
}

```

```
}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customization

The **fill** and **border** properties are used to customize the background color and border of the tooltip, respectively. The **textStyle** property in the tooltip is used to customize the font of the tooltip text. Additionally, the **highlightColor** property can be used to change the color of the data point when hovering.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    [highlightColor]="highlightColor" [highlightMode]='highlightMode'
    [tilt]='tilt' [legendSettings]="legendSettings" [tooltip]='tooltip'>
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'>
        </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
  })
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public title?: string;
  public legendSettings?: Object;
  public tilt?: number;
  public tooltip?: object;
  public highlightColor?: string;
  public highlightMode?: string;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 13 },
      { x: 'Feb', y: 13 },
      { x: 'Mar', y: 17 },
      { x: 'Apr', y: 13.5 }
    ]
  }
}
```

```

];
this.legendSettings = { visible: false };
this.tooltip = {
  enable: true,
  format: '${series.name} ${point.x} : ${point.y}',
  fill: '#7bb4eb',
  border: {
    width: 2,
    color: 'grey'
  }
}
this.tilt = -45;
this.highlightColor = 'Red';
this.highlightMode = 'Point';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customization of individual tooltip

Using the `tooltipRender` event, you can customize tooltip values for a particular point.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent, CircularChart3DTooltipRenderEventArgs }
from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    (tooltipRender)="tooltipRender($event)" [tilt]='tilt'
    [legendSettings]="legendSettings" [tooltip]='tooltip'>
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'>
        </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
  })
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public title?: string;
  public legendSettings?: Object;

```



```

public tilt?: number;
public tooltip?: object;
public tooltipRender: Function | any;
@ViewChild('chart')
public chartObj?: CircularChart3DComponent;
ngOnInit(): void {
    this.dataSource = [
        { x: 'Jan', y: 13 },
        { x: 'Feb', y: 13 },
        { x: 'Mar', y: 17 },
        { x: 'Apr', y: 13.5 }
    ];
    this.legendSettings = { visible: false };
    this.tooltip = { enable: true, format: '${point.x} : <b>${point.y}%</b>' };
    this.tilt = -45;
    this.tooltipRender = (args: CircularChart3DTooltipRenderEventArgs) => {
        if (args.point.index === 3) {
            args.text = args.point.x + ' ' + ':' + args.point.y + ' ' + ' ' +
            'customtext';
            args.textStyle!.color = '#f48042';
        }
        else {
            args.textStyle!.color = 'White';
        }
    }
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Title and subtitle in Angular 3D Circular Chart component

Title

The 3D Circular Chart can be given a title by using the `title` property to display information about the plotted data.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',

```

```

template: `
  <ejs-circularchart3d #chart style='display:block;' align='center'
  [title]='title' [tilt]='tilt' [legendSettings]="legendSettings">
    <e-circularchart3d-series-collection>
      <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
    >
      </e-circularchart3d-series></e-circularchart3d-series-collection>
    </ejs-circularchart3d>`
  ))
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public tilt?: number;
  public title?: string;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Saudi Arabia', y: 58 },
      { x: 'Persian Gulf', y: 15 },
      { x: 'Canada', y: 13 },
      { x: 'Venezuela', y: 8 },
      { x: 'Mexico', y: 3 },
      { x: 'Russia', y: 2 },
      { x: 'Miscellaneous', y: 1 }
    ];
    this.legendSettings = { visible: false };
    this.title = 'Oil and other liquid imports in USA',
    this.tilt = -45;
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Title customization

The title of the 3D Circular Chart can be customized using the `titleStyle` property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,

```

```

    selector: 'app-container',
    template: `
      <ejs-circularchart3d #chart style='display:block;' align='center'
      [title]='title' [titleStyle]='titleStyle' [tilt]='tilt'
      [legendSettings]="legendSettings">
        <e-circularchart3d-series-collection>
          <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
        >
          </e-circularchart3d-series></e-circularchart3d-series-collection>
        </ejs-circularchart3d>`
  })
  export class AppComponent implements OnInit {
    public dataSource?: Object[];
    public legendSettings?: Object;
    public tilt?: number;
    public title?: string;
    public titleStyle?: object;
    @ViewChild('chart')
    public chartObj?: CircularChart3DComponent;
    ngOnInit(): void {
      this.dataSource = [
        { x: 'Saudi Arabia', y: 58 },
        { x: 'Persian Gulf', y: 15 },
        { x: 'Canada', y: 13 },
        { x: 'Venezuela', y: 8 },
        { x: 'Mexico', y: 3 },
        { x: 'Russia', y: 2 },
        { x: 'Miscellaneous', y: 1 }
      ];
      this.legendSettings = { visible: false };
      this.title = 'Oil and other liquid imports in USA',
      this.tilt = -45;
      this.titleStyle = {
        fontFamily: "Arial",
        fontStyle: 'italic',
        fontWeight: 'regular',
        color: "#E27F2D",
        size: '23px'
      };
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Subtitle

The 3D Circular Chart can be given a subtitle by using the `subTitle` property to display information about the plotted data.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    [title]='title' [subTitle]='subTitle' [tilt]='tilt'
    [legendSettings]="legendSettings">
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
      >
        </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
  })
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public tilt?: number;
  public title?: string;
  public subTitle?: string;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Saudi Arabia', y: 58 },
      { x: 'Persian Gulf', y: 15 },
      { x: 'Canada', y: 13 },
      { x: 'Venezuela', y: 8 },
      { x: 'Mexico', y: 3 },
      { x: 'Russia', y: 2 },
      { x: 'Miscellaneous', y: 1 }
    ];
    this.legendSettings = { visible: false };
    this.title = 'Oil and other liquid imports in USA',
    this.subTitle = 'In the year 2014 - 2015',
    this.tilt = -45;
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Subtitle customization

The subtitle of the 3D Circular Chart can be customized using the `subTitleStyle` property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-circularchart3d #chart style='display:block;' align='center'
    [title]='title' [subTitle]='subTitle' [subTitleStyle]='subTitleStyle'
    [tilt]='tilt' [legendSettings]="legendSettings">
      <e-circularchart3d-series-collection>
        <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
      >
        </e-circularchart3d-series></e-circularchart3d-series-collection>
      </ejs-circularchart3d>`
  })
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public legendSettings?: Object;
  public tilt?: number;
  public title?: string;
  public subTitle?: string;
  public subTitleStyle?: object;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Saudi Arabia', y: 58 },
      { x: 'Persian Gulf', y: 15 },
      { x: 'Canada', y: 13 },
      { x: 'Venezuela', y: 8 },
      { x: 'Mexico', y: 3 },
      { x: 'Russia', y: 2 },
      { x: 'Miscellaneous', y: 1 }
    ];
    this.legendSettings = { visible: false };
    this.title = 'Oil and other liquid imports in USA',
    this.subTitle = 'In the year 2014 - 2015',
    this.tilt = -45;
    this.subTitleStyle = {
      fontFamily: 'Arial',
      fontStyle: 'italic',
      fontWeight: 'regular',
      color: '#E27F2D',
      size: '13px'
    };
  }
};
```

```
}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Print and Export in Angular 3D Circular Chart component

Print

The rendered 3D Circular Chart can be printed directly from the browser by calling the public method **print**. The ID of the 3D Circular Chart div element must be passed as the input parameter to that method.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<button id='print' (click)='print()'>Print</button>
  <ejs-circularchart3d #chart style='display:block;' align='center'
  [tilt]='tilt' [legendSettings]="legendSettings">
    <e-circularchart3d-series-collection>
      <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
  [radius]='radius'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
    </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public title?: string;
  public legendSettings?: Object;
  public tilt?: number;
  public radius?: string;
  @ViewChild('chart')
  public chartObj?: CircularChart3DComponent;
  ngOnInit(): void {
    this.dataSource = [
      { x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 },
      { x: 'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
      { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 },
      { x: 'Jul', y: 26 }, { x: 'Aug', y: 25 },
      { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 },
      { x: 'Nov', y: 15 }, { x: 'Dec', y: 15 }
    ]
  }
}
```

```

];
this.legendSettings = { visible: false };
this.tilt = -45;
this.radius = '100%';
}
print() {
  this.chartObj?.print();
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Export

The rendered 3D Circular Chart can be exported to JPEG, PNG, or SVG format using the `export` method. Additionally, you can export the 3D Circular Chart as a PDF format using the `pdfExport` method. The input parameters for this method are `type` for the format and `fileName` for the result.

Input parameters for this method are `Export Type` for `format` and `fileName` of result.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularChart3DAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewChild } from '@angular/core';
import { CircularChart3DComponent } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    CircularChart3DAllModule
  ],
  providers: [CircularChart3DAllModule],
  standalone: true,
  selector: 'app-container',
  template: `<button id='export' (click)='export()'>Export</button>
<ejs-circularchart3d #chart style='display:block;' align='center'
[tilt]='tilt' [legendSettings]="legendSettings"
[enableExport]='enableExport'>
  <e-circularchart3d-series-collection>
    <e-circularchart3d-series [dataSource]='dataSource' xName='x' yName='y'
[radius]='radius'>
    </e-circularchart3d-series></e-circularchart3d-series-collection>
  </ejs-circularchart3d>`
})
export class AppComponent implements OnInit {
  public dataSource?: Object[];
  public title?: string;
  public legendSettings?: Object;
  public tilt?: number;
  public radius?: string;
  public enableExport?: boolean;

```

```

@ViewChild('chart')
public chartObj?: CircularChart3DComponent;
ngOnInit(): void {
  this.dataSource = [
    { x: 'Jan', y: 3 }, { x: 'Feb', y: 3.5 },
    { x: 'Mar', y: 7 }, { x: 'Apr', y: 13.5 },
    { x: 'May', y: 19 }, { x: 'Jun', y: 23.5 },
    { x: 'Jul', y: 26 }, { x: 'Aug', y: 25 },
    { x: 'Sep', y: 21 }, { x: 'Oct', y: 15 },
    { x: 'Nov', y: 15 }, { x: 'Dec', y: 15 }
  ];
  this.legendSettings = { visible: false };
  this.tilt = -45;
  this.radius = '100%';
  this.enableExport = true;
}
export() {
  this.chartObj?.circularChartExport3DModule.export('PDF',
'circular3DChart')
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Circular Gauge

Getting started with Angular Circular gauge component

This section explains you the steps required to create a simple circular gauge and demonstrate the basic usage of circular gauge control.

Setup Angular Environment

You can use [Angular CLI](#) to setup your Angular applications.

To install Angular CLI use the following command.

```
`bash
```

```
npm install -g @angular/cli
```

```
`
```

Create an Angular Application

Start a new Angular application using below Angular CLI command.

```
`bash
```

```
ng new my-app
```

```
cd my-app
```

```
`
```


Installing Syncfusion Circular Gauge package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(`>=20.2.36`) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-circulargauge](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-circulargauge --save
```

```
,
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-circulargauge@ngcc](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-circulargauge@ngcc --save
```

```
,
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
```

```
@syncfusion/ej2-angular-circulargauge:"20.2.38-ngcc"
```

```
,
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Registering Circular Gauge Module

Import Circular Gauge module into Angular application(`app.module.ts`) from the package `@syncfusion/ej2-angular-circulargauge` [`src/app/app.module.ts`].

```
`typescript
```

```
import { NgModule } from '@angular/core';
```

```
import { BrowserModule } from '@angular/platform-browser';
```

```
// import the CircularGaugeModule for the Circular Gauge component
```

```
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge';
```

```
import { AppComponent } from './app.component';
@NgModule({
  //declaration of ChartModule into NgModule
  imports: [ BrowserModule, CircularGaugeModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

- Modify the template in `app.component.ts` file to render the `ej2-angular-circulargauge` component

[src/app/app.component.ts].

```
`javascript
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  selector: 'app-container',
  // specifies the template string for the Charts component
  template: <ejs-circulargauge id='circular-container'></ejs-circulargauge>,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent { }
```

<!-- markdownlint-disable MD033 -->

Now use the `<code>app-container</code>` in the index.html instead of default one.

```
<app-container></app-container>
```

- Now run the application in the browser using the below command.

```
npm start
```

The below example shows a basic Circular Gauge.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GaugeTooltipService } from '@syncfusion/ej2-angular-circulargauge'
import { Component } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [ GaugeTooltipService ],
  standalone: true,
  selector: 'app-container',
  // specifies the template string for the CircularGauge component
  template: `<ejs-circulargauge id="circular-container"></ejs-
circulargauge>`
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Set Pointer Value

Pointer value is used to indicate the gauge value using [value](#) property in [pointers](#).

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GaugeTooltipService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [ GaugeTooltipService ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis>
          <e-pointers>
            <e-pointer value=35></e-pointer>
          </e-pointers>
        </e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
```

```
export class AppComponent implements OnInit {
  ngOnInit(): void {
    // Initialize objects.
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Gauge dimensions in Angular Circular gauge component**Size for Container**

You can set width and height to the element of the container. It determines the gauge size. Also, you can set the size via inline or CSS as demonstrated below.

```
<div id='container'>
<div id='circular-container' style="width:650px; height:350px;"></div>
</div>
```

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  ngOnInit(): void {
    // Initialize objects.
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

<!-- markdownlint-disable MD036 -->

Size for Circular Gauge

<!-- markdownlint-disable MD036 -->

You can also set size for the gauge directly through [width](#) and [height](#) properties.

In Pixel

You can set the size of the gauge in pixel as demonstrated below.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container" width="650" height="350">`
    `</ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  ngOnInit(): void {
    // Initialize objects.
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

In Percentage

By setting value in percentage, gauge gets its dimension with respect to its container. For example, when the height is '50%', gauge renders to half of the container height.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
```

```

    selector: 'app-container',
    template:
      `<ejs-circulargauge id="circular-container" width="80%" height="50%">
      </ejs-circulargauge>`
  })
  export class AppComponent implements OnInit {
    ngOnInit(): void {
      // Initialize objects.
    }
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note: When you do not specify the size, it takes 450px as the height and window size as its width.

Gauge axes in Angular Circular gauge component

By default, gauge will be displayed with an axis.

Each axis contains its own ranges, pointers and annotation.

<!-- markdownlint-disable MD036 -->

Axis Customization

You can customize the width and color of an axis line by using [lineStyle](#) property. Background for an axis can be customized by using [background](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis [lineStyle]="lineStyle" background="#b2d8d8"></e-axis>
      </e-axes>
    </ejs-circulargauge>`
  })
  export class AppComponent implements OnInit {
    public lineStyle?: Object;
    ngOnInit(): void {
      // Initialize objects
      this.lineStyle= {
        width: 2,

```

```

        color: 'red'
    };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

Angles and Direction

Circular gauge axis can sweep from 0 to 360 degrees. By default start angle of an axis is 200 degree and end angle is 160 degree and you can customize this option by using [startAngle](#) and [endAngle](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis startAngle= 270 endAngle=90></e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  ngOnInit(): void {
    // Initialize objects
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

The [direction](#) property enables you to render the gauge axis either in **ClockWise** or in **AntiClockWise** direction.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis direction="AntiClockWise"></e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  ngOnInit(): void {
    // Initialize objects
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

<!-- markdownlint-disable MD036 -->

Axis Radius

By default, radius of an axis is calculated based on the available size. You can customize this, by using [radius](#) property.

It takes value either in **percentage** or in **pixel**.

In Pixel

You can set the radius of the gauge in pixel as demonstrated below,

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
```



```

    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis radius="150"></e-axis>
      </e-axes>
    </ejs-circulargauge>`
  })
  export class AppComponent implements OnInit {
    ngOnInit(): void {
      // Initialize objects
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

In Percentage

By setting value in percentage, gauge gets its dimension with respect to its available size.

For example, when the radius is '50%', gauge renders to half of the available size.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis radius="50%"></e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  ngOnInit(): void {
    // Initialize objects
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';

```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

<!-- markdownlint-disable MD036 -->

Ticks

You can customize the [height](#), [color](#) and [width](#) of major and minor ticks by using [majorTicks](#) and [minorTicks](#) property.

By default, [interval](#) for [majorTicks](#) will be calculated automatically and also you can customize the interval for major and minor ticks using [interval](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Tick Position

Both minor and major ticks can be moved by using [offset](#) and [position](#) property. The [offset](#) defines the distance between the axis and ticks. By default, offset value is 0. The [position](#) will place the ticks either inside or outside of the axis.

By default, ticks will be placed **inside** the axis.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis [majorTicks]="majorTicks" [minorTicks]="minorTicks"></e-
axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  public majorTicks?: Object;
  public minorTicks?: Object;
  ngOnInit(): void {
    // Initialize objects
    this.majorTicks= {
      interval: 10,
      color:'red',
      height: 10,
      width: 3,
      position: 'Inside',
      offset: 5
    };
    this.minorTicks= {
      interval: 5,
      color:'green',
      height: 5,
      width: 2,
      position: 'Inside',
      offset: 5
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Labels

Labels of an axis can be customized by using [font](#) property in [labelStyle](#) options.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis [labelStyle]="labelStyle"></e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  public labelStyle?: Object;
  ngOnInit(): void {
    // Initialize objects
    this.labelStyle= {
      font: {
        color: 'red',
        size: '20px',
        fontWeight: 'Bold'
      }
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Label Position

Labels can be moved by using [offset](#) or [position](#) property.

The [offset](#) defines the distance between the labels and ticks.

By default, offset value is 0.

The [position](#) will place the labels either inside or outside of the axis.

By default, labels will be placed **inside** the axis.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis [labelStyle]="labelStyle"></e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  public labelStyle?: Object;
  ngOnInit(): void {
    // Initialize objects
    this.labelStyle= {
      position: 'Outside',
      offset: 5
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Auto Angle

Labels can be swept along the axis angle by enabling [autoAngle](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
```

```

    template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis [labelStyle]="labelStyle"></e-axis>
      </e-axes>
    </ejs-circulargauge>`
  })
  export class AppComponent implements OnInit {
    public labelStyle?: Object;
    ngOnInit(): void {
      // Initialize objects
      this.labelStyle = {
        autoAngle: true
      };
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Smart Labels

When an axis makes a complete circle, then the first and last label of the axis will get overlap with each other.

In this scenario, you can either hide 1st or last label using [hiddenLabel](#) property.

When [hiddenLabel](#) value is **First**, then the 1st label will be hidden and when the [hiddenLabel](#) value is **Last**, then the last label will be hidden.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis minimum=0 maximum=12 startAngle=0 endAngle=360
        [majorTicks]="majorTicks" [minorTicks]="minorTicks"
        [labelStyle]="labelStyle"></e-axis>
      </e-axes>
    </ejs-circulargauge>`
  })
  export class AppComponent implements OnInit {

```

```

public majorTicks?: Object;
public minorTicks?: Object;
public labelStyle?: Object;
ngOnInit(): void {
    // Initialize objects
    this.majorTicks= {
        interval: 1,
        position: 'Inside',
        height: 10
    };
    this.minorTicks= {
        interval: 0.2,
        position: 'Inside',
        height: 5
    };
    this.labelStyle= {
        position: 'Inside',
        hiddenLabel: 'First'
    };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Label Format

Axis labels can be formatted by using [format](#) property in [labelStyle](#) and its supports all globalize format.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis [labelStyle]="labelStyle"></e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  public labelStyle?: Object;
  ngOnInit(): void {
    // Initialize objects
  }
}

```

```

        this.labelStyle= {
            format: 'p1'
        };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

The following table describes the result of applying some commonly used label formats on numeric values.

<!-- markdownlint-disable MD033 -->

Label Value	Label Format property value	Result	Description
1000	n1	1000.0	The Number is rounded to 1 decimal place
1000	n2	1000.00	The Number is rounded to 2 decimal place
1000	n3	1000.000	The Number is rounded to 3 decimal place
0.01	p1	1.0%	The Number is converted to percentage with 1 decimal place
0.01	p2	1.00%	The Number is converted to percentage with 2 decimal place
0.01	p3	1.000%	The Number is converted to percentage with 3 decimal place
1000	c1	\$1,000.0	The Currency symbol is appended to number and number is rounded to 1 decimal place
1000	c2	\$1,000.00	The Currency symbol is appended to number and number is rounded to 2 decimal place

Custom Label Format

Axis labels support custom label format using placeholder like {value}°C, in which the value represent the axis label e.g 20°C.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [

```



```

        CircularGaugeModule
    ],
    standalone: true,
    selector: 'app-container',
    template:
`<ejs-circulargauge id="circular-container">
    <e-axes>
        <e-axis [labelStyle]="labelStyle"></e-axis>
    </e-axes>
</ejs-circulargauge>`
})
export class AppComponent implements OnInit {
    public labelStyle?: Object;
    ngOnInit(): void {
        // Initialize objects
        this.labelStyle= {
            format: '{value}°C'
        };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Hide intersecting axis labels

When the axis labels overlap with each other, you can hide the intersected labels by setting the `hideIntersectingLabel` property to true in the axis.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
    imports: [
        CircularGaugeModule
    ],
    standalone: true,
    selector: 'app-container',
    template:
`<ejs-circulargauge id="circular-container">
    <e-axes>
        <e-axis minimum=0 maximum=200 startAngle=270 endAngle=90
hideIntersectingLabel=true [majorTicks]="majorTicks"
[minorTicks]="minorTicks"></e-axis>
    </e-axes>
</ejs-circulargauge>`
})
export class AppComponent implements OnInit {
    public minorTicks?: Object;
}

```

```

public majorTicks?: Object;
ngOnInit(): void {
    // Initialize objects
    this.majorTicks= {
        interval: 4,
    };
    this.minorTicks= {
        interval: 2,
    };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Minimum and Maximum

The [minimum](#) and [maximum](#) properties enables you to customize the start and end values of an axis.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis minimum=50 maximum=250></e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  ngOnInit(): void {
    // Initialize objects
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Multiple Axes

In addition to the default axis, you can add n number of axis to a gauge.

Each axis will have its own ranges, pointers, annotations and customization options.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Gauge ranges in Angular Circular gauge component

You can categories certain interval on gauge axis using [ranges](#) property.

Start and End

Start and end value of a range in an axis can be customized by using [start](#) and [end](#) properties.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { AnnotationsService, GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [AnnotationsService, GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis>
          <e-ranges>
            <e-range start=40 end=80></e-range>
          </e-ranges>
        </e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  ngOnInit(): void {
    // Initialize objects.
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customization

Color and thickness of the range can be customized by using [color](#), [startWidth](#) and [endWidth](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { AnnotationsService, GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [AnnotationsService, GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis minimum=0 maximum=100 [majorTicks]="majorTicks"
[minorTicks]="minorTicks">
          <e-ranges>
            <e-range start=40 end=80 startWidth=15 endWidth=15
color="#ff5985"></e-range>
          </e-ranges>
        </e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  public majorTicks?: Object;
  public minorTicks?: Object;
  ngOnInit(): void {
    // Initialize objects.
    this.majorTicks = {
      interval: 10
    };
    this.minorTicks = {
      interval: 5
    };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

Radius

You can place the range inside or outside of the axis by using [radius](#) property.

The radius of the range can takes value either in percentage or in pixels.

By default, ranges take 100% of the axis radius.

In Pixel

You can set the radius of the range in pixel as demonstrated below,

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { AnnotationsService, GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [AnnotationsService, GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

<!-- markdownlint-disable MD036 -->

In Percentage

By setting value in percentage, range gets its dimension with respect to its axis radius.

For example, when the radius is '50%', range renders to half of the axis radius.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { AnnotationsService, GradientService } from '@syncfusion/ej2-angular-circulargauge'
```

```
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [AnnotationsService, GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis>
          <e-ranges>
            <e-range start=40 end=80 radius='50%'></e-range>
          </e-ranges>
        </e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  ngOnInit(): void {
    // Initialize objects.
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

<!-- markdownlint-disable MD010 -->

Dragging Range

The ranges can be dragged on the axis line by clicking and dragging the same. To enable or disable the range drag, use the [enableRangeDrag](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container" enableRangeDrag='true'
    height='250px' width='250px'>
      <e-axes>
```

```

        <e-axis>
            <e-pointers>
                <e-pointer value=50></e-pointer>
            </e-pointers>
            <e-ranges>
                <e-range start=0 end=100 startWidth=8 endWidth=8
color="#30B32D" radius='108%'></e-range>
            </e-ranges>
        </e-axis>
    </e-axes>
</ejs-circulargauge>`
    })
    export class AppComponent implements OnInit {
        public animation?: Object;
        ngOnInit(): void { }
    }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Multiple Ranges

You can add multiple ranges to an axis with the above customization as demonstrated below.

Note: You can set the range color to axis ticks and labels by enabling `useRangeColor` property in [majorTicks](#), [minorTicks](#) and [labelStyle](#) object.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge';
import { AnnotationsService, GradientService } from '@syncfusion/ej2-angular-circulargauge';
import { Component, OnInit } from '@angular/core';
@Component({
    imports: [
        CircularGaugeModule
    ],
    providers: [AnnotationsService, GradientService],
    standalone: true,
    selector: 'app-container',
    template:
`<ejs-circulargauge id="circular-container">
    <e-axes>
        <e-axis [majorTicks]="majorTicks" [minorTicks]="minorTicks"
[labelStyle]="labelStyle">
            <e-ranges>
                <e-range start=0 end=25 radius='108%'></e-range>
                <e-range start=25 end=50 radius='70%'></e-range>
                <e-range start=50 end=75 radius='70%'></e-range>
                <e-range start=75 end=100 radius='108%'></e-range>
            </e-ranges>
        </e-axis>
    </e-axes>
</ejs-circulargauge>`
})

```



```

        </e-ranges>
      </e-axis>
    </e-axes>
  </ejs-circulargauge>`
  })
  export class AppComponent implements OnInit {
    public majorTicks?: Object;
    public minorTicks?: Object;
    public labelStyle?: Object;
    ngOnInit(): void {
      // Initialize objects.
      this.majorTicks = {
        useRangeColor: true
      };
      this.minorTicks = {
        useRangeColor: true
      };
      this.labelStyle = {
        useRangeColor: true
      };
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Rounded corner radius

You can customize the corner radius using the `roundedCornerRadius` property in `ranges`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge';
import { AnnotationsService, GradientService } from '@syncfusion/ej2-angular-circulargauge';
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [AnnotationsService, GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis>
          <e-ranges>
            <e-range start=40 end=80 radius='50%'
              roundedCornerRadius=6></e-range>

```

```

        </e-ranges>
      </e-axis>
    </e-axes>
  </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  ngOnInit(): void {
    // Initialize objects.
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Gradient Color

Gradient support allows to add multiple colors in the ranges and pointers of the circular gauge. The following gradient types are supported in the circular gauge.

- Linear Gradient
- Radial Gradient

Linear Gradient

Using linear gradient, colors will be applied in a linear progression. The start value of the linear gradient will be set using the [startValue](#) property. The end value of the linear gradient will be set using the [endValue](#) property. The color stop values such as color, opacity and offset are set using [colorStop](#) property.

To apply linear gradient to the range, follow the below code sample.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { AnnotationsService, GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [AnnotationsService, GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge style='display:block;' [title]='title' centerY='57%'
    [titleStyle]='titleStyle'>
      <e-axes>

```

```

        <e-axis [lineStyle]='lineStyle' radius='90%' startAngle=200
endAngle=130 minimum=0 maximum=14 [labelStyle]='labelStyle'
[majorTicks]='majorTicks'
        [minorTicks]='minorTicks' [ranges]='ranges'>
        <e-pointers>
            <e-pointer type='Marker' [value]=12
markerShape='Image'
imageUrl='https://ej2.syncfusion.com/angular/demos/assets/circular-
gauge/images/foot-ball.png' radius='108%'
                [markerWidth]=28 [markerHeight]=28
[animation]='animation1'>
            </e-pointer>
            <e-pointer type='Marker' [value]=11
markerShape='Image'
imageUrl='https://ej2.syncfusion.com/angular/demos/assets/circular-
gauge/images/basket-ball.png' radius='78%'
                [markerWidth]=28 [markerHeight]=28
[animation]='animation2'>
            </e-pointer>
            <e-pointer type='Marker' [value]=10
markerShape='Image'
imageUrl='https://ej2.syncfusion.com/angular/demos/assets/circular-
gauge/images/golf-ball.png' radius='48%'
                [markerWidth]=28 [markerHeight]=28
[animation]='animation3'>
            </e-pointer>
            <e-pointer type='Marker' [value]=12
markerShape='Image'
imageUrl='https://ej2.syncfusion.com/angular/demos/assets/circular-
gauge/images/athletics.png' radius='0%'
                [markerWidth]=90 [markerHeight]=90
[animation]='animation4'>
            </e-pointer>
            <e-pointer type='Marker' [value]=0.1
markerShape='Image'
imageUrl='https://ej2.syncfusion.com/angular/demos/assets/circular-
gauge/images/girl.png' radius='108%'
                [markerWidth]=28 [markerHeight]=28
[animation]='animation1'>
            </e-pointer>
            <e-pointer type='Marker' [value]=0.1
markerShape='Image'
imageUrl='https://ej2.syncfusion.com/angular/demos/assets/circular-
gauge/images/man-one.png' radius='78%' [markerWidth]=28
                [markerHeight]=28 [animation]='animation1'>
            </e-pointer>
            <e-pointer type='Marker' [value]=0.1
markerShape='Image'
imageUrl='https://ej2.syncfusion.com/angular/demos/assets/circular-
gauge/images/man-two.png' radius='48%' [markerWidth]=28
                [markerHeight]=28 [animation]='animation1'>
            </e-pointer>
        </e-pointers>
        <e-annotations>
            <e-annotation content='12 M' radius='108%' angle=98
zIndex='1'> </e-annotation>

```

```

        <e-annotation content='11 M' radius='80%' angle=81
zIndex='1'> </e-annotation>
        <e-annotation content='10 M' radius='50%' angle=69
zIndex='1'> </e-annotation>
        <e-annotation content='Doe' radius='108%' angle=190
zIndex='1'> </e-annotation>
        <e-annotation content='Almaida' radius='80%'
angle=185 zIndex='1'> </e-annotation>
        <e-annotation content='John' radius='50%' angle=180
zIndex='1'> </e-annotation>
    </e-annotations>
  </e-axis>
</e-axes>
</ejs-circulargauge>
,
})
export class AppComponent implements OnInit {
  public ranges?: Object[];
  public titleStyle?: Object;
  public title?: string;
  public animation1?: Object;
  public animation2?: Object;
  public animation3?: Object;
  public animation4?: Object;
  public lineStyle?: Object;
  public labelStyle?: Object;
  public majorTicks?: Object;
  public minorTicks?: Object;
  public rangeLinearGradient: Object = {
    startValue: '0%', endValue: '100%',
    colorStop: [
      { color: '#9E40DC', offset: '0%', opacity: 0.9 },
      { color: '#E63B86', offset: '70%', opacity: 0.9 }
    ]
  };
  ngOnInit(): void {
    // Initialize objects.
    this.ranges = [{
      start: 0, end: 12, radius: '115%',
      startWidth: 25, endWidth: 25,
      linearGradient : this.rangeLinearGradient
    }, {
      start: 0, end: 11, radius: '85%',
      startWidth: 25, endWidth: 25,
      linearGradient : this.rangeLinearGradient
    }, {
      start: 0, end: 10, radius: '55%',
      startWidth: 25, endWidth: 25,
      linearGradient : this.rangeLinearGradient
    }
  ];
    this.titleStyle = { size: '18px' };
    this.title = 'Short Put Distance';
    this.animation1 = { duration: 1500 };
    this.animation2 = { duration: 1200 };
    this.animation3 = { duration: 900 };
    this.animation4 = { duration: 0 };
    this.lineStyle = { width: 0 };
    this.labelStyle = { font: { size: '0px' } };
  }
}

```

```

        this.majorTicks = { width: 0 };
        this.minorTicks = { width: 0 };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Radial Gradient

Using radial gradient, colors will be applied in circular progression. The inner circle position of the radial gradient will be set using the [innerPosition](#) property. The outer circle position of the radial gradient can be set using the [outerPosition](#) property. The color stop values such as color, opacity and offset are set using [colorStop](#) property.

To apply radial gradient to the range, follow the below code sample.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge';
import { AnnotationsService, GradientService } from '@syncfusion/ej2-angular-circulargauge';
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [AnnotationsService, GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge style='display:block;' [title]='title' centerY='57%'
    [titleStyle]='titleStyle'>
      <e-axes>
        <e-axis [lineStyle]='lineStyle' radius='90%' startAngle=200
        endAngle=130 minimum=0 maximum=14 [labelStyle]='labelStyle'
        [majorTicks]='majorTicks'
          [minorTicks]='minorTicks' [ranges]='ranges'>
          <e-pointers>
            <e-pointer type='Marker' [value]=12
            markerShape='Image'
            imageUrl='https://ej2.syncfusion.com/angular/demos/assets/circular-
            gauge/images/foot-ball.png' radius='108%'
              [markerWidth]=28 [markerHeight]=28
            [animation]='animation1'>
              </e-pointer>
            <e-pointer type='Marker' [value]=11
            markerShape='Image'
            imageUrl='https://ej2.syncfusion.com/angular/demos/assets/circular-
            gauge/images/basket-ball.png' radius='78%'

```

```

                                [markerWidth]=28 [markerHeight]=28
[animation]='animation2'>
                                </e-pointer>
                                <e-pointer type='Marker' [value]=10
markerShape='Image'
imageUrl='https://ej2.syncfusion.com/angular/demos/assets/circular-
gauge/images/golf-ball.png' radius='48%'
                                [markerWidth]=28 [markerHeight]=28
[animation]='animation3'>
                                </e-pointer>
                                <e-pointer type='Marker' [value]=12
markerShape='Image'
imageUrl='https://ej2.syncfusion.com/angular/demos/assets/circular-
gauge/images/athletics.png' radius='0%'
                                [markerWidth]=90 [markerHeight]=90
[animation]='animation4'>
                                </e-pointer>
                                <e-pointer type='Marker' [value]=0.1
markerShape='Image'
imageUrl='https://ej2.syncfusion.com/angular/demos/assets/circular-
gauge/images/girl.png' radius='108%'
                                [markerWidth]=28 [markerHeight]=28
[animation]='animation1'>
                                </e-pointer>
                                <e-pointer type='Marker' [value]=0.1
markerShape='Image'
imageUrl='https://ej2.syncfusion.com/angular/demos/assets/circular-
gauge/images/man-one.png' radius='78%' [markerWidth]=28
                                [markerHeight]=28 [animation]='animation1'>
                                </e-pointer>
                                <e-pointer type='Marker' [value]=0.1
markerShape='Image'
imageUrl='https://ej2.syncfusion.com/angular/demos/assets/circular-
gauge/images/man-two.png' radius='48%' [markerWidth]=28
                                [markerHeight]=28 [animation]='animation1'>
                                </e-pointer>
                                </e-pointers>
                                <e-annotations>
                                    <e-annotation content='12 M' radius='108%' angle=98
zIndex='1'> </e-annotation>
                                    <e-annotation content='11 M' radius='80%' angle=81
zIndex='1'> </e-annotation>
                                    <e-annotation content='10 M' radius='50%' angle=69
zIndex='1'> </e-annotation>
                                    <e-annotation content='Doe' radius='108%' angle=190
zIndex='1'> </e-annotation>
                                    <e-annotation content='Almaida' radius='80%'
angle=185 zIndex='1'> </e-annotation>
                                    <e-annotation content='John' radius='50%' angle=180
zIndex='1'> </e-annotation>
                                </e-annotations>
                                </e-axis>
                                </e-axes>
                                </ejs-circulargauge>
,
    })
    export class AppComponent implements OnInit {

```

```

public ranges?: Object[];
public titleStyle?: Object;
public title?: string;
public animation1?: Object;
public animation2?: Object;
public animation3?: Object;
public animation4?: Object;
public lineStyle?: Object;
public labelStyle?: Object;
public majorTicks?: Object;
public minorTicks?: Object;
public rangeRadialGradient: Object = {
  radius: '50%', innerPosition: { x: '50%', y: '50%' },
  outerPosition: { x: '50%', y: '50%' },
  colorStop: [
    { color: '#9E40DC', offset: '90%', opacity: 0.9 },
    { color: '#E63B86', offset: '160%', opacity: 0.9 }
  ]
};
ngOnInit(): void {
  // Initialize objects.
  this.ranges = [{
    start: 0, end: 12, radius: '115%',
    startWidth: 25, endWidth: 25,
    radialGradient: this.rangeRadialGradient
  }, {
    start: 0, end: 11, radius: '85%',
    startWidth: 25, endWidth: 25,
    radialGradient: this.rangeRadialGradient
  }, {
    start: 0, end: 10, radius: '55%',
    startWidth: 25, endWidth: 25,
    radialGradient: this.rangeRadialGradient
  }];
  this.titleStyle = { size: '18px' };
  this.title = 'Short Put Distance';
  this.animation1 = { duration: 1500 };
  this.animation2 = { duration: 1200 };
  this.animation3 = { duration: 900 };
  this.animation4 = { duration: 0 };
  this.lineStyle = { width: 0 };
  this.labelStyle = { font: { size: '0px' } };
  this.majorTicks = { width: 0 };
  this.minorTicks = { width: 0 };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See also

- [Tooltip for Ranges](#)

Gauge pointers in Angular Circular Gauge Component

Pointers are used to indicate values on the axis. Value of the pointer can be modified using the [value](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Gauge supports 3 types of pointers such as **Needle**, **RangeBar** and **Marker**.

You can choose any one of the pointer by using [type](#) property.

Needle Pointers

A needle pointer contains three parts, a needle, a cap / knob and a tail.

The length of the needle can be customized by using [radius](#) property.

The length of the tail can be customized by using [length](#) property.

The radius of the cap can be customized by using [radius](#) in cap object.

The needle and tail length takes value either in [percentage](#) or [pixel](#).

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis>
          <e-pointers>
            <e-pointer value=90 radius="50%" [cap]="cap"
[needleTail]="needleTail"></e-pointer>
          </e-pointers>
        </e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  public cap?: Object;
  public needleTail?: Object;
  ngOnInit(): void {
    // Initialize objects
    this.cap= {
      radius: 10
    };
    this.needleTail= {
      length: '25%'
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

<!-- markdownlint-disable MD036 -->

Customization

Needle color and width can be customized by using [color](#) and [pointerWidth](#) property.

Cap and tails can be customized by using [cap](#) and [needleTail](#) object.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

<!-- markdownlint-disable MD010 -->

The appearance of the needle pointer can be customized by using [needleStartWidth](#) and [needleEndWidth](#).

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis startAngle=270 endAngle=90 [lineStyle]="lineStyle"
[labelStyle]="labelStyle" [majorTicks]="majorTicks" [minorTicks]="minorTicks"
        radius='90%' minimum=0 maximum=100>
          <e-pointers>
            <e-pointer value=70 radius="80%" pointerWidth=2
color='green' [needleStartWidth]="needleStartWidth"
[needleEndWidth]="needleEndWidth" [cap]="cap" [needleTail]="needleTail"
[animation]="animation"></e-pointer>
          </e-pointers>
          <e-annotations>
            <e-annotation angle=180 radius="20%" zIndex=1>
              <ng-template #content>
                <div>
                  <div
style="color:#757575; font-family:Roboto; font-size:14px;padding-top:
26px">Customized Needle</div>
                </div>
              </ng-template>
            </e-annotation>
          </e-annotations>
        </e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  public cap?: Object;
  public needleTail?: Object;
  public animation?: Object;
  public labelStyle?: Object;
  public lineStyle?: Object;
  public majorTicks?: Object;
  public minorTicks?: Object;
  public needleStartWidth?: Number;
  public needleEndWidth?: Number;
  ngOnInit(): void {
```

```

// Initialize objects
    this.needleStartWidth = 4;
    this.needleEndWidth = 4;
    this.animation = {
        enable: true , duration: 1000
    };
    this.cap= {
        radius: 8,
        color: 'green'
    };
    this.needleTail= {
        length: '0%'
    };
    this.labelStyle= {
        position: 'Outside',
        font:{
            size: '0px', color: '#1E7145'
        }
    };
    this.lineStyle= {
        width: 3, color: '#1E7145'
    };
    this.majorTicks = {
        width: 1,
        height: 0,
        interval: 100
    };
    this.minorTicks = {
        height: 0,
        width: 0
    };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

RangeBar Pointer

RangeBar pointer is like ranges in an axis, that can be placed on gauge to mark the pointer value.

RangeBar starts from the beginning of the gauge and ends at the pointer value.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule

```

```

    ],
    providers: [GradientService],
    standalone: true,
    selector: 'app-container',
    template:
      `<ejs-circulargauge id="circular-container">
        <e-axes>
          <e-axis>
            <e-pointers>
              <e-pointer value=50 type="RangeBar" radius="60%"></e-
pointer>
            </e-pointers>
          </e-axis>
        </e-axes>
      </ejs-circulargauge>`
  ))
  export class AppComponent implements OnInit {
    ngOnInit(): void {
      // Initialize objects
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customization

RangeBar can be customized in terms of color, border and thickness by using [color](#), [border](#) and [pointerWidth](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis>
          <e-pointers>
            <e-pointer value=50 type="RangeBar" radius="60%"
pointerWidth=15 color='#007DD1' [border]="pointerBorder"></e-pointer>
          </e-pointers>
        </e-axis>
      </e-axes>
    </ejs-circulargauge>`
})

```

```

        </e-axis>
      </e-axes>
    </ejs-circulargauge>`
  })
  export class AppComponent implements OnInit {
    public pointerBorder?: Object;
    ngOnInit(): void {
      // Initialize objects
      this.pointerBorder = {
        color: 'grey',
        width: 2
      };
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD010 -->

Rounded corner for range bar pointer

The start and end pointers of range bar in the circular gauge are rounded to form arc gauges.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `

```

```

    })
    export class AppComponent implements OnInit {
      public lineStyle?: Object;
      ngOnInit(): void {
        // Initialize objects
        this.lineStyle = {
          color: 'transparent'
        };
      }
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Marker Pointer

Different type of marker shape can be used to mark the pointer value in axis. You can change the marker shape using [markerShape](#)

property in pointer.

Gauge supports the below marker shape.

- Circle
- Rectangle
- Triangle
- InvertedTriangle
- Diamond

We can use image instead of rendering marker shape to denote the pointer value. It can be achieved by setting [markerShape](#) to Image and assigning image path to [imageUrl](#) in pointer.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge';
import { GradientService } from '@syncfusion/ej2-angular-circulargauge';
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis>
          <e-pointers>

```

```

        <e-pointer value=90 type="Marker"
markerShape="InvertedTriangle" radius="100%" [markerHeight]="markerSize"
[markerWidth]="markerSize"></e-pointer>
    </e-pointers>
</e-axis>
</e-axes>
</ejs-circulargauge>`
))
export class AppComponent implements OnInit {
    public markerSize?: number;
    ngOnInit(): void {
        // Initialize objects
        this.markerSize = 15;
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customization

The marker can be customized in terms of color, border, width and height by using [color](#), [border](#), [markerWidth](#) and [markerHeight](#) property in [pointer](#).

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis>
          <e-pointers>
            <e-pointer value=90 type="Marker" markerShape="Triangle"
radius="100%" color="white" [markerHeight]="markerSize"
[markerWidth]="markerSize" [border]="pointerBorder"></e-pointer>
          </e-pointers>
        </e-axis>
      </e-axes>
    </ejs-circulargauge>`
  })
export class AppComponent implements OnInit {

```



```

    public pointerBorder?: Object;
    public markerSize?: number;
    ngOnInit(): void {
        // Initialize objects
        this.markerSize = 15;
        this.pointerBorder= {
            color: '#007DD1',
            width: 2
        };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD010 -->

Dragging Pointer

The pointers can be dragged over the axis line by clicking and dragging the same. To enable or disable the pointer drag, use the [enablePointerDrag](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Multiple Pointers

In addition to the default pointer, you can add n number of pointer to an axis by using **pointers** property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis>
          <e-pointers>
            <e-pointer value=90 type="Marker"
markerShape="InvertedTriangle" radius="100%" [markerHeight]="markerSize"
[markerWidth]="markerSize"></e-pointer>
            <e-pointer value=90 type="RangeBar" radius="60%"
pointerWidth=10></e-pointer>
            <e-pointer value=90 radius="60%" pointerWidth=25
[cap]="pointerCap" [needleTail]= "pointerTail"></e-pointer>
          </e-pointers>
        </e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  public markerSize?: number;
  public pointerCap?: Object;
  public pointerTail?: Object;
  ngOnInit(): void {
    // Initialize objects
    this.markerSize = 15;
    this.pointerCap = {
      radius: 15,
      border: {
        width: 5
      }
    }
  }
}
```

```

    };
    this.pointerTail = {
        length: '22%'
    };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Animation

Pointer will get animate on loading the gauge, this can be handled by using [animation](#) property in pointer.

The [enable](#) property in animation allows you to enable or disable the animation.

The [duration](#) property specify the duration of the animation in milliseconds.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge';
import { GradientService } from '@syncfusion/ej2-angular-circulargauge';
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `

```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Gradient Color

Gradient support allows to add multiple colors in the ranges and pointers of the circular gauge. The following gradient types are supported in the circular gauge.

- Linear Gradient
- Radial Gradient

Linear Gradient

Using linear gradient, colors will be applied in a linear progression. The start value of the linear gradient will be set using the [startValue](#) property. The end value of the linear gradient will be set using the [endValue](#) property. The color stop values such as color, opacity and offset are set using [colorStop](#) property.

The linear gradient can be applied to all pointer types like marker, range bar and needle. To do so, follow the below code sample.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `

```

```

public pointers?: Object[];
public pointerLinearGradient: Object = {
  startValue: '0%',
  endValue: '100%',
  colorStop: [
    { color: '#FEF3F9', offset: '0%', opacity: 0.9 },
    { color: '#E63B86', offset: '70%', opacity: 0.9 }]
};
ngOnInit(): void {
  // Initialize objects
  this.lineStyle= {
    width: 3, color: '#E63B86'
  };
  this.labelStyle = {
    font: { size: '0px' }
  };
  this.majorTicks = {
    height: 0
  };
  this.minorTicks = {
    height: 0
  };
  this.pointers = [{
    radius: '80%',
    value: 80,
    animation: { enable: true, duration: 1000 },
    pointerWidth: 10,
    linearGradient: this.pointerLinearGradient,
    cap: {
      radius: 8,
      color: 'white',
      border: {
        color: '#E63B86',
        width: 1
      }
    },
    needleTail: {
      length: '20%',
      linearGradient: this.pointerLinearGradient
    }
  }, {
    radius: '60%', value: 40,
    animation: { duration: 1000 },
    pointerWidth: 10,
    linearGradient: this.pointerLinearGradient,
    cap: {
      radius: 8, color: 'white',
      border: { color: '#E63B86', width: 1 }
    },
    needleTail: {
      length: '20%',
      linearGradient: this.pointerLinearGradient
    }
  }
  ]};
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Radial Gradient

Using radial gradient, colors will be applied in circular progression. The inner circle position of the radial gradient will be set using the [innerPosition](#) property. The outer circle position of the radial gradient can be set using the [outerPosition](#) property. The color stop values such as color, opacity and offset are set using [colorStop](#) property.

The radial gradient can be applied to all pointer types like marker, range bar and needle. To do so, follow the below code sample.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GradientService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis startAngle=270 endAngle=90 radius='90%' minimum=0
maximum=100 [lineStyle]='lineStyle' [labelStyle]='labelStyle'
[majorTicks]='majorTicks' [minorTicks]='minorTicks' [pointers]='pointers'>
          </e-axis>
        </e-axes>
      </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  public lineStyle?: Object;
  public labelStyle?: Object;
  public majorTicks?: Object;
  public minorTicks?: Object;
  public pointers?: Object[];
  public pointerRadialGradient : Object = {
    radius: '50%',
    innerPosition: { x: '50%', y: '50%' },
    outerPosition: { x: '50%', y: '50%' },
    colorStop: [
      { color: '#FEF3F9', offset: '0%', opacity: 0.9 },
      { color: '#E63B86', offset: '60%', opacity: 0.9 }
    ]
  };
};
```

```

ngOnInit(): void {
  // Initialize objects
  this.lineStyle= {
    width: 3, color: '#E63B86'
  };
  this.labelStyle = {
    font: { size: '0px'}
  };
  this.majorTicks = {
    height: 0
  };
  this.minorTicks = {
    height: 0
  };
  this.pointers = [{
    radius: '80%',
    value: 80,
    animation: { enable: true, duration: 1000 },
    pointerWidth: 10,
    radialGradient: this.pointerRadialGradient,
    cap: {
      radius: 8,
      color: 'white',
      border: {
        color: '#E63B86',
        width: 1
      }
    },
    needleTail: {
      length: '20%',
      radialGradient: this.pointerRadialGradient
    }
  }, {
    radius: '60%', value: 40,
    animation: { duration: 1000 },
    pointerWidth: 10,
    radialGradient: this.pointerRadialGradient,
    cap: {
      radius: 8, color: 'white',
      border: { color: '#E63B86', width: 1 }
    },
    needleTail: {
      length: '20%',
      radialGradient: this.pointerRadialGradient
    }
  }
  ]};
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Gauge annotations in Angular Circular gauge component

<!-- markdownlint-disable MD010 -->

Annotations are used to mark a specific area of interest in the gauge with texts, shapes or images.

Content

You can place any custom element on the axis area by assigning the id of the element to [content](#) property of [annotation](#) object.

Note: To use annotation feature, we need to inject `AnnotationsService` into the `NgModule.providers`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { AnnotationsService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [AnnotationsService],
  standalone: true,
  selector: 'app-container',
  template:
    `

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```



```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Position

Annotation can be placed around the axis by using [radius](#) and [angle](#) property.

For example, if the angle is 90 degree and the radius is 110%, then the annotation, will be placed at the right side of the axis.

Radius of the annotation takes value either in pixel or percentage.

By setting value in percentage, annotation gets its position with respect to its axis radius.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { AnnotationsService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [AnnotationsService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis>
          <e-pointers>
            <e-pointer value = 50></e-pointer>
          </e-pointers>
          <e-annotations>
            <e-annotation angle=90 radius="150%" zIndex="1">
              <ng-template #content>
                <div>
                  <div><span>Pointer Value : 50</span>
                </div>
              </ng-template>
            </e-annotation>
          </e-annotations>
        </e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  ngOnInit(): void {
    // Initialize objects
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Sub Gauge

As the annotation allows you to place any custom element, we can initialize a gauge to the element and can be used to place that in another gauge.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { AnnotationsService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
import { CircularGauge } from '@syncfusion/ej2-angular-circulargauge';
import { ILoadedEventArgs } from '@syncfusion/ej2-circulargauge';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [AnnotationsService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container" (loaded)='loaded!($event)'>
      <e-axes>
        <e-axis minimum=0 maximum=12 startAngle=0 endAngle=360
[lineStyle]="lineStyle" [labelStyle]="labelStyle">
          <e-annotations>
            <e-annotation angle=270 radius='40%' zIndex='1'>
              <ng-template #content>
                <div id="subGauge"
style="width:90px;height:90px"></div>
              </ng-template>
            </e-annotation>
            <e-annotation angle=90 radius='40%' zIndex='1'>
              <ng-template #content>
                <div id="subTime" style="width:90px;height:90px">
                  <span>6:30 PM</span>
                </div>
              </ng-template>
            </e-annotation>
          </e-annotations>
          <e-ranges>
            <e-range start=0 end=3 color="rgba(29,29,29,0.7)"></e-
range>
            <e-range start=3 end=12
color="rgba(168,145,102,0.1)"></e-range>
          </e-ranges>
          <e-pointers>
            <e-pointer pointerWidth=5 radius="40%" value=6.5
color="#1d1d1d" [border]='pointerBorder' [cap]='pointerCap'
[needleTail]='pointerTail' [animation]='pointerAnimation'></e-pointer>
```

```

        <e-pointer pointerWidth=5 radius="60%" value=6
color="#1d1d1d" [border]='pointerBorder' [cap]='pointerCap'
[needleTail]='pointerTail' [animation]='pointerAnimation'></e-pointer>
        <e-pointer pointerWidth=5 radius="70%" value=9.8
color="#a89166" [cap]='pointerCap1' [needleTail]='pointerTail1'
[animation]='pointerAnimation'></e-pointer>
    </e-pointers>
</e-axis>
</e-axes>
</ejs-circulargauge>`
}))
export class AppComponent implements OnInit {
    public labelStyle?: Object;
    public lineStyle?: Object;
    public pointerBorder?: Object;
    public pointerCap?: Object;
    public pointerTail?: Object;
    public pointerAnimation?: Object;
    public pointerCap1?: Object;
    public pointerTail1?: Object;
    public loaded?: Function;
    ngOnInit(): void {
        // Initialize objects
        this.labelStyle = {
            hiddenLabel: 'First'
        };
        this.lineStyle = {
            width: 0
        };
        this.pointerBorder= { width: 1, color: 'rgb(29,29,29)' };
        this.pointerCap= {
            color: 'rgb(29,29,29)',
            radius: 0,
            border: {
                width: 0.2,
                color: 'red'
            }
        };
        this.pointerTail= {
            length: '0%'
        };
        this.pointerAnimation= {
            enable: false
        };
        this.pointerCap1= {
            color: 'rgba(168,145,102,1)',
            radius: 4,
            border: {
                width: 0.2,
                color: 'rgba(168,145,102,1)'
            }
        };
        this.pointerTail1= {
            color: 'rgba(168,145,102,1)',
            length: '20%'
        };
        this.loaded = (args: ILoadedEventArgs): void=> {

```

```

    let gauge: CircularGauge = new CircularGauge({
      axes: [{
        minimum: 0,
        maximum: 12,
        startAngle: 0,
        endAngle: 360,
        majorTicks: { interval: 3 },
        lineStyle: { width: 0 },
        ranges: [{
          start: 0, end: 3,
          startWidth: 5, endWidth: 5,
          color: 'rgba(29,29,29,0.7)'
        }, {
          start: 3, end: 12,
          startWidth: 5, endWidth: 5,
          color: 'rgba(168,145,102,0.1)'
        }],
        labelStyle: {
          hiddenLabel: 'First',
          offset: -5
        },
      },
      pointers: [{
        pointerWidth: 2,
        radius: '40%',
        color: 'rgb(29,29,29)',
        border: { width: 1, color: 'rgb(29,29,29)' },
        cap: {
          color: 'rgb(29,29,29)',
          radius: 2,
          border: {
            width: 0.2,
            color: 'red'
          }
        },
      },
      needleTail: {
        length: '0%'
      },
      animation: {
        enable: false
      }
    }],
  }, '#subGauge');
}
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See also

- [Tooltip for Annotation](#)

Animation in Angular Circular Gauge component

All of the elements in the Circular Gauge, such as the axis lines, ticks, labels, ranges, pointers, and annotations, can be animated sequentially by using the [animationDuration](#) property. The animation for the Circular Gauge is enabled when the `animationDuration` property is set to an appropriate value in milliseconds, providing a smooth rendering effect for the component. If the `animationDuration` property is set to `0`, which is the default value, the animation effect is disabled. If the animation is enabled, the component will behave in the following order.

1. The axis line will be animated in the rendering direction (clockwise or anticlockwise).
2. Each tick line and label will then be animated.
3. If available, ranges will be animated.
4. If available, pointers will be animated in the same way as [pointer animation](#).
5. If available, annotations will be animated.

The animation of the Circular Gauge is demonstrated in the following example.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule, AnnotationsService, GradientService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GradientService, AnnotationsService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-circulargauge style='display:block;' animationDuration=2000
>
  <e-axes>
    <e-axis radius='80%' startAngle=230 endAngle=130
[majorTicks]='majorTicks' [minorTicks]='minorTicks'
[lineStyle]='lineStyle' [labelStyle]='labelStyle'
[ranges]="ranges">
    <e-pointers>
      <e-pointer value=60 radius='60%' pointerWidth=7
color='#c06c84' [animation]='animation' [cap]="cap"
[needleTail]="tail">
    </e-pointer>
  </e-pointers>
  <e-annotations>
    <e-annotation angle="165" radius="35%" zIndex='1'>
      <ng-template #content>
        <div>
          <div style="font-size:18px;margin-left: -
20px;margin-top: -12px; color:#9DD55A">60</div>
```

```

        </div>
      </ng-template>
    </e-annotation>
  </e-annotations>
</e-axis>
</e-axes>
</ejs-circulargauge>`,
  })
export class AppComponent implements OnInit {
  public animation?: Object;
  public majorTicks?: Object;
  public minorTicks?: Object;
  public lineStyle?: Object;
  public labelStyle?: Object;
  public cap?: Object;
  public tail?: Object;
  public ranges?: Object;
  ngOnInit(): void {
    this.majorTicks = {
      offset: 5,
    };
    this.minorTicks = {
      offset: 5,
    };
    this.ranges = [
      {
        start: 0,
        end: 30,
        color: '#E63B86',
        startWidth: 22,
        endWidth: 22,
        radius: '60%',
        linearGradient: {
          startValue: '0%',
          endValue: '100%',
          colorStop: [
            { color: '#9e40dc', offset: '0%', opacity: 1 },
            { color: '#d93c95', offset: '70%', opacity: 1 },
          ],
        },
      },
      {
        start: 30,
        end: 60,
        color: '#E0E0E0',
        startWidth: 22,
        endWidth: 22,
        radius: '60%',
      },
    ];
    this.lineStyle = {
      width: 8,
      color: '#E0E0E0',
    };
    this.labelStyle = {
      font: {
        fontFamily: 'inherit',

```

```

    },
    offset: -1,
  };
  this.animation = {
    enable: true,
    duration: 500,
  };
  this.cap = {
    radius: 8,
    color: '#c06c84',
    border: { width: 0 },
  };
  this.tail = {
    length: '0%',
  };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Only the pointer of the Circular Gauge can be animated individually, not the axis lines, ticks, labels, ranges, and annotations. You can refer this [link](#) to enable only pointer animation.

Gauge legend in Angular Circular gauge component

Legend provides valuable information for interpreting what the circular gauge axis range displays, and they can be represented in various colors, shapes, and other identifiers based on the data. It gives a breakdown of what each symbol represents in the axis range of circular gauge.

You can add the legend for circular gauge ranges by setting the visible property of `legendSettings` to true.

<!-- markdownlint-disable MD036 -->

Legend customization

Customization option is also provided for the legend shape, alignment, and position.

Position and alignment

The position of the legend is used to place legend in various positions. You can use the `position` property in `legendSettings`. Based on the position, the legend item will be aligned. The following options are available to customize the legend position:

- Top
- Bottom
- Left
- Right
- Custom
- Auto

The legend alignment is used to align the legend items in specific location. You can use the alignment property in `legendSettings` to align the legend items. The following options are available to customize the legend alignment:

- Near
- Center
- Far

The legends can also be positioned to absolute position using the `location.x` and `location.y` properties available in `legendSettings`.

Legend size

The legend size can be modified using the `height` and `width` properties in `legendSettings`.

Legend opacity

To specify the transparency for legend shape, set the `opacity` property in `legendSettings`.

Legend shape

To change the legend item shape, specify the desired `shape` in the shape property of the legend. By default, the shape of the legend is `circle`.

It also supports the following shapes:

- Circle
- Rectangle
- Diamond
- Triangle
- InvertedTriangle
- Image

You can customize a shape using the `shapeWidth` and `shapeHeight` properties.

Legend padding

You can control the spacing between the legend items using the `padding` option of the legend. The default value of padding is 5.

Legend border

You can customize the legend border using the `border` option in the legend. The legend border can be customized using the border `color` and `width` properties.

Font of the legend text

The `font` of the legend item text can be customized using the following properties:

- `fontFamily`
- `fontStyle`
- `fontWeight`
- `opacity`
- `color`
- `size`

The following code example shows how to add legend in the gauge.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { LegendService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [LegendService],
  standalone: true,
  selector: 'app-container',
  template:
    `

```

```
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

<!-- markdownlint-disable MD036 -->

Toggle option in legend

The toggle option has been provided for legend. So, if you toggle the legend, the given color will be changed to the corresponding circular gauge range. You can enable the toggle option using `toggleVisibility` in the `legendSettings` property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge';
import { LegendService } from '@syncfusion/ej2-angular-circulargauge';
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [LegendService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container"
    [legendSettings]='legendSettings'>
      <e-axes>
        <e-axis minimum=0 maximum=100 [majorTicks]="majorTicks"
        [minorTicks]="minorTicks" [lineStyle]="lineStyle">
          <e-ranges>
            <e-range start=0 end=25 radius='108%'></e-range>
            <e-range start=25 end=50 radius='108%'></e-range>
            <e-range start=50 end=75 radius='108%'></e-range>
            <e-range start=75 end=100 radius='108%'></e-range>
          </e-ranges>
        </e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  public lineStyle?: Object;
  public majorTicks?: Object;
  public minorTicks?: Object;
  public legendSettings?: object;
  ngOnInit(): void {
    // Initialize objects.
```

```

        this.lineStyle = {
            useRangeColor: true
        };
        this.majorTicks = {
            useRangeColor: true
        };
        this.minorTicks = {
            useRangeColor: true
        };
        this.legendSettings= {
            visible: true,
            toggleVisibility: true
        };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

Paging support in legend

By default, paging will be enabled if the legend items exceed the legend bounds. You can view each legend item by navigating between the pages using navigation buttons.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { LegendService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [LegendService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container"
    [legendSettings]='legendSettings'>
      <e-axes>
        <e-axis minimum=0 maximum=100 [majorTicks]="majorTicks"
        [minorTicks]="minorTicks" [lineStyle]="lineStyle">
          <e-ranges>
            <e-range start=0 end=25 radius='108%'></e-range>
            <e-range start=25 end=50 radius='108%'></e-range>
            <e-range start=50 end=75 radius='108%'></e-range>
            <e-range start=75 end=100 radius='108%'></e-range>
          </e-ranges>
        </e-axis>
      </e-axes>
    `
})

```

```

        </e-axes>
    </ejs-circulargauge>`
    })
    export class AppComponent implements OnInit {
        public lineStyle?: Object;
        public majorTicks?: Object;
        public minorTicks?: Object;
        public legendSettings?: object;
        ngOnInit(): void {
            // Initialize objects.
            this.lineStyle = {
                useRangeColor: true
            };
            this.majorTicks = {
                useRangeColor: true
            };
            this.minorTicks = {
                useRangeColor: true
            };
            this.legendSettings= {
                visible: true,
                height: '50'
            };
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

Legend text customization

You can customize the legend text using `legendText` property in `ranges`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { LegendService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [LegendService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container"
    [legendSettings]='legendSettings'>
    <e-axes>

```

```

        <e-axis minimum=0 maximum=100 [majorTicks]="majorTicks"
[minorTicks]="minorTicks" [lineStyle]="lineStyle">
            <e-ranges>
                <e-range start=0 end=25 radius='108%'
legendText='light air'></e-range>
                <e-range start=25 end=50 radius='108%'
legendText='light air'></e-range>
                <e-range start=50 end=75 radius='108%'
legendText='light breeze'></e-range>
                <e-range start=75 end=100 radius='108%'
legendText='light breeze'></e-range>
            </e-ranges>
        </e-axis>
    </e-axes>
</ejs-circulargauge>`
    })
    export class AppComponent implements OnInit {
        public lineStyle?: Object;
        public majorTicks?: Object;
        public minorTicks?: Object;
        public legendSettings?: object;
        ngOnInit(): void {
            // Initialize objects.
            this.lineStyle= {
                useRangeColor: true
            };
            this.majorTicks = {
                useRangeColor: true
            };
            this.minorTicks = {
                useRangeColor: true
            };
            this.legendSettings= {
                visible: true,
            };
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

legendRendering event will be triggered before rendering each legend item, using this event you can customize needed legend items using following arguments.

Argument Name	Description
---------------	-------------

---	---
-----	-----

fill	Specifies the legend shape color
------	----------------------------------

|text| Specifies the current legend text |

|shape| Customize the shape of the legends |

|name| Specifies the name of the event |

|cancel| Set to true, to cancel the event status |

Gauge user interaction in Angular Circular gauge component

Tooltip for pointers

Circular gauge will displays the pointer details through [tooltip](#), when the mouse is moved over the pointer.

<!-- markdownlint-disable MD036 -->

Enable Tooltip

By default, tooltip is not visible. Enable the tooltip by setting [enable](#) property to true and by injecting GaugeTooltipService into the NgModule.providers.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GaugeTooltipService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GaugeTooltipService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container" [tooltip]="tooltip">
      <e-axes>
        <e-axis>
          <e-pointers>
            <e-pointer value=70></e-pointer>
          </e-pointers>
        </e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  public tooltip?: Object;
  ngOnInit(): void {
    // Initialize objects
    this.tooltip = {
      enable: true
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

<!-- markdownlint-disable MD036 -->

Template

Any HTML elements can be displayed in the tooltip by using the [template](#) property of the tooltip.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GaugeTooltipService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GaugeTooltipService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container" [tooltip]="tooltip">
      <e-axes>
        <e-axis>
          <e-pointers>
            <e-pointer value=70></e-pointer>
          </e-pointers>
        </e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  public tooltip?: Object;
  ngOnInit(): void {
    // Initialize objects
    this.tooltip = {
      enable: true,
      template: '<div id="templateWrap"><div style="float: right;
padding-left:10px; line-height:30px;"><span>Pointer &#160;&#160;:&#160;
${value}</span></div></div>'
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Tooltip for ranges

Circular gauge displays the information about the ranges through tooltip when hovering the mouse over the ranges. You can enable this feature by setting the type property of tooltip to 'Range' in the array collection.

Tooltip customization for ranges

To customize the range tooltip, use the `rangeSettings` property in tooltip. The following options are available to customize the range tooltip:

- `fill` - Specifies the range tooltip fill color.
- `textStyle` - Specifies the range tooltip text style.
- `format` - Specifies the range content format.
- `template` - Specifies the custom template for tooltip.
- `enableAnimation` - Animates as it moves from one point to another.
- `border` - Specifies the tooltip border.
- `showMouseAtPosition` - Displays the position of the tooltip on the cursor position.

Tooltip for annotations

Circular gauge displays the information about the annotations through tooltip when hovering the mouse over the annotation. You can enable this feature by setting the type property of tooltip to 'Annotation' in the array collection.

Tooltip customization for annotations

To customize the annotation tooltip, use the `annotationSettings` property in tooltip. The following options are available to customize the annotation tooltip:

- `fill` - Specifies the annotation tooltip fill color.
- `textStyle` - Specifies the annotation tooltip text style.
- `format` - Specifies the annotation content format.
- `template` - Specifies the tooltip content with custom template.
- `enableAnimation` - Animates as it moves from one point to another.
- `border` - Specifies the tooltip border.

The following code example shows the tooltip for the pointers, ranges and annotations.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GaugeTooltipService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';

@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GaugeTooltipService],
  standalone: true,
  selector: 'app-container',
```



```

    template:
      `<ejs-circulargauge id="circular-container" [tooltip]="tooltip"
enablePointerDrag=true>
    <e-axes>
      <e-axis minimum=0 maximum=120 radius="90%" startAngle= 240
endAngle=120 [majorTicks]="majorTicks" [minorTicks]="minorTicks"
[lineStyle]="lineStyle" [annotations]='annotaions'>
        <e-pointers>
          <e-pointer value=70 radius="60%" [cap]="cap"></e-pointer>
        </e-pointers>
        <e-ranges>
          <e-range start=0 end=50 radius='102%' startWidth=10
endWidth=10 color='#3A5DC8'></e-range>
          <e-range start=50 end=120 radius='102%' startWidth=10
endWidth=10 color='#33BCBD'></e-range>
        </e-ranges>
      </e-axis>
    </e-axes>
  </ejs-circulargauge>`
  })
  export class AppComponent implements OnInit {
    public lineStyle?: Object;
    public majorTicks?: Object;
    public minorTicks?: Object;
    public tooltip?: Object;
    public cap?: Object;
    public annotaions : any;
    ngOnInit(): void {
      // Initialize objects
      this.tooltip = {
        type:['Pointer', 'Range', 'Annotation'],
        enable: true,
        enableAnimation: false,
        annotationSettings: { template:'<div>CircularGauge</div>' },
        rangeSettings: { fill:'red' }
      };
      this.lineStyle= {
        width: 0
      };
      this.majorTicks = {
        color: 'white', offset: -5, height: 12
      };
      this.minorTicks = {
        width: 0
      };
      this.annotaions = [{
        content: 'CircularGauge',
        angle: 180, zIndex: '1',
      }];
      this.cap= {
        radius: 10,
        border: {
          color: '#33BCBD',
          width: 5
        }
      };
    }
  }

```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Pointer Drag

Pointers can be dragged over the axis value. This can be achieved by clicking and dragging the pointer.

To enable or disable the pointer drag, you can use [enablePointerDrag](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { GaugeTooltipService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [GaugeTooltipService],
  standalone: true,
  selector: 'app-container',
  template:
    `

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Gauge print and export in Angular Circular gauge component

Print

To use the print functionality, we should inject the `PrintService` into the `@NgModule.providers` and set the `allowPrint` property to `true`. The rendered circular gauge can be printed directly from the browser by calling the method `print`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit, ViewChild } from '@angular/core';
import { PrintService, CircularGaugeComponent } from '@syncfusion/ej2-
angular-circulargauge';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Export

Image Export

To use the image export functionality, we should inject the `ImageExportService` into the `@NgModule.providers` and set the `allowImageExport` property to `true`. The rendered circular gauge

can be exported as an image using the [export](#) method. The method requires two parameters: image type and file name. The circular gauge can be exported as an image in the following formats.

- JPEG
- PNG
- SVG

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit, ViewChild } from '@angular/core';
import { ImageExportService, CircularGaugeComponent } from '@syncfusion/ej2-angular-circulargauge';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container" [allowImageExport]=true
#gauge>
    </ejs-circulargauge><div> <button id='export'
(click)='export()'>Export</button></div>`,
  providers: [ImageExportService]
})
export class AppComponent implements OnInit {
  @ViewChild('gauge')
  public gaugeObj?: CircularGaugeComponent;
  ngOnInit(): void {
    // ngOnInit code here
  }
  public export(): void {
    this.gaugeObj!.print();
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

We can get the image file as base64 string for the JPEG and PNG formats. The circular gauge can be exported to image as a base64 string using the [export](#) method. There are four parameters required: image type, file name, orientation of the exported PDF document which must be set as **null** for image export and finally **allowDownload** which should be set as **false** to return base64 string.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
```

```
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit, ViewChild } from '@angular/core';
import { ImageExportService, CircularGaugeComponent } from '@syncfusion/ej2-
angular-circulargauge';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container" [allowImageExport]=true
#gauge>
</ejs-circulargauge>
<div><button id='export' (click)='export()'>Export</button></div>`,
  providers: [ImageExportService]
})
export class AppComponent implements OnInit {
  @ViewChild('gauge')
  public gaugeObj?: CircularGaugeComponent;
  ngOnInit(): void {
  }
  public export() {
    const promise = this.gaugeObj!.export('PNG', 'Gauge', undefined, false);
    promise.then((data) => {
      document.writeln(data);
    })
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

PDF Export

To use the PDF export functionality, we should inject the PdfExportService into the @NgModule.providers and set the allowPdfExport property to true. The rendered circular gauge can be exported as PDF using the export method. The export method requires three parameters: file type, file name and orientation of the PDF document. The orientation setting is optional and "0" indicates portrait and "1" indicates landscape.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit, ViewChild } from '@angular/core';
import { PdfExportService, CircularGaugeComponent } from '@syncfusion/ej2-
angular-circulargauge';
@Component({
  imports: [
```

```

        CircularGaugeModule
    ],
    standalone: true,
    selector: 'app-container',
    template:
`<ejs-circulargauge id="circular-container" [allowPdfExport]=true #gauge>
</ejs-circulargauge><div> <button id='export'
(click)='export()'>Export</button></div>`,
    providers: [PdfExportService]
})
export class AppComponent implements OnInit {
    @ViewChild('gauge')
    public gaugeObj?: CircularGaugeComponent;
    ngOnInit(): void {
    }
    public export() {
        this.gaugeObj!.export('PDF', 'Gauge', 0);
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note: The exporting of the circular gauge as base64 string is not supported in the PDF export.

Gauge appearance in Angular Circular gauge component

Gauge Title

Circular gauge can be given a title by using [title](#) property, to show the information about the gauge.

Title can be customized by using [titleStyle](#) property in gauge.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
    imports: [
        CircularGaugeModule
    ],
    standalone: true,
    selector: 'app-container',
    template:
`<ejs-circulargauge id="circular-container" title="Speedometer"
[titleStyle]="titleStyle">
</ejs-circulargauge>`
})
export class AppComponent implements OnInit {
    public titleStyle?: Object;
    ngOnInit(): void {
    }
}

```

```

        // Initialize objects.
        this.titleStyle = {
            color: '#27d5ff'
        };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Gauge Position

<!-- markdownlint-disable MD036 -->

Gauge can be positioned anywhere in the container with the help of [centerX](#) and [centerY](#) property and it accepts values either in percentage or in pixels.

The default value of the [centerX](#) and [centerY](#) property is 50%, which means gauge will get rendered to the centre of the container.

In Pixel

You can set the mid point of the gauge in pixel as demonstrated below,

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container" centerX='20' centerY='20'>
      <e-axes>
        <e-axis [lineStyle]="lineStyle" startAngle=90 endAngle=180>
        </e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  public lineStyle?: Object;
  ngOnInit(): void {
    // Initialize objects.
    this.lineStyle = {
      width: 2,
      color: '#F8F8F8'
    };
  }
}

```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

In Percentage

By setting the value in percentage, gauge gets its mid point with respect to its plot area.

For example, when the [centerX](#) value as '0%' and [centerY](#) value is '50%', gauge will get positioned at the top left corner of the plot area.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container" centerX='10%' centerY='50%'>
      <e-axes>
        <e-axis [lineStyle]="lineStyle" startAngle=0 endAngle=180>
        </e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  public lineStyle?: Object;
  ngOnInit(): void {
    // Initialize objects.
    this.lineStyle = {
      width: 2,
      color: '#F8F8F8'
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```



```
<!-- markdownlint-disable MD036 -->
```

Area Customization

Customize the gauge background

Using [background](#) and [border](#) properties, you can change the background color and border of the circular gauge.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Gauge Margin

You can set margin for gauge from its container through [margin](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `

```

```

        this.minorTicks = { width: 1, color: '#8c8c8c' };
        this.lineStyle = { width: 2 };
        this.margin = { left: 40, right: 40, top: 40, bottom: 40 };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Radius calculation based on angles

Render semi or quarter circular gauges by modifying the start and end angles. By enabling the radius based on angle option, the radius of circular gauge will be calculated based on the start and end angles to avoid excess white space.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container" centerX='10%' centerY='50%'>
      <e-axes>
        <e-axis [lineStyle]="lineStyle" startAngle=270 endAngle=90>
        </e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  public lineStyle?: Object;
  ngOnInit(): void {
    // Initialize objects.
    this.lineStyle = {
      width: 2,
      color: '#F8F8F8'
    };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Accessibility in Angular Circular gauge component

The Circular Gauge component follows commonly used accessibility guidelines and standards, such as [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#).

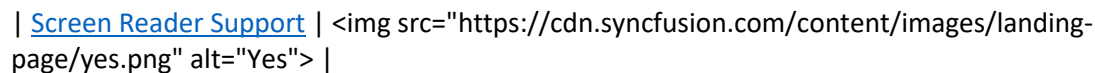
The accessibility compliance for the Circular Gauge component is outlined below.

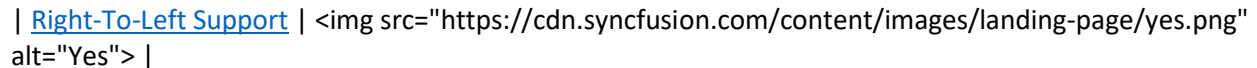
| Accessibility Criteria | Compatibility |

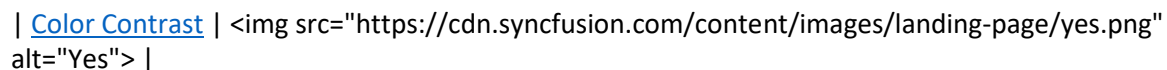
| -- | -- |

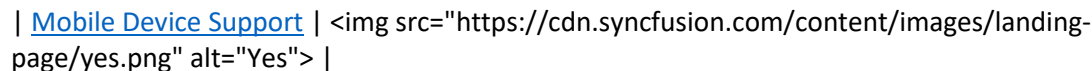
| [WCAG 2.2 Support](#) |  |

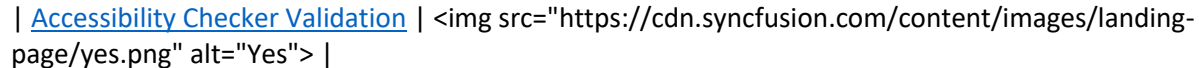
| [Section 508 Support](#) |  |

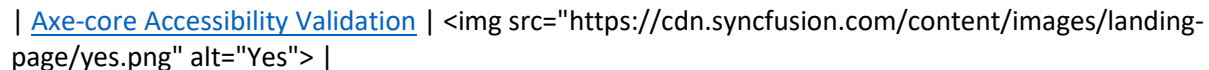
| [Screen Reader Support](#) |  |

| [Right-To-Left Support](#) |  |

| [Color Contrast](#) |  |

| [Mobile Device Support](#) |  |

| [Accessibility Checker Validation](#) |  |

| [Axe-core Accessibility Validation](#) |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Circular Gauge component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Circular Gauge component:

| Attributes | Purpose |

| --- | --- |

| **role=region** | It is specified in the pointer, annotation, and title. The pointer supports the interactive drag-and-drop function to update the pointer value. |

| **aria-label** | Provides an accessible name for the axis labels, title, legend item labels, text pointers and annotation. |

Screen reading in Circular Gauge

Accessibility in the Circular Gauge component ensures that all users, regardless of ability or disability, can use screen reading. The following Circular Gauge elements will be read aloud using screen reading software, such as Narrator for Windows.

| Elements | Description |

| --- | --- |

| Axis labels | Reads the axis labels of the Circular Gauge. |

| Title | Reads the title of the Circular Gauge. |

| Legend item label | Reads the label of the legend item in the Circular Gauge. |

| Text pointer | Reads the text content shown as a pointer in Circular Gauge. |

| Annotation | Reads the content specified in the annotation. |

Ensuring accessibility

The Circular Gauge component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Circular Gauge component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Circular Gauge component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Internationalization in Angular Circular Gauge component

Circular Gauge provides internationalization support for below elements.

- Axis Labels
- Tooltip

For more information about number formatter, you can refer [internationalization](#).

Globalization

Globalization is the process of designing and developing a component that works in different cultures/locales.

Internationalization library is used to globalize number in Circular Gauge using [format](#) property in [labelStyle](#).

```
<!-- markdownlint-disable MD036 -->
```

Numeric Format

In the below example, axis labels are globalized to **EUR**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { loadCldr, L10n, setCulture, setCurrencyCode } from '@syncfusion/ej2-base'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-circulargauge id="circular-container">
      <e-axes>
        <e-axis [labelStyle]="labelStyle"></e-axis>
      </e-axes>
    </ejs-circulargauge>`
})
export class AppComponent implements OnInit {
  public labelStyle?: Object;
  ngOnInit(): void {
    // Initialize objects.
    this.labelStyle = {
      // Label format set as currency.
      format: 'c'
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Right-to-left

Circular Gauge can render its elements from right to left, which improves the user experience for certain language users. To do so, set the [enableRtl](#) property to **true**. When this property is enabled, elements such as the tooltip and legend will be rendered from right to left. Meanwhile, the axis can be rendered from right to left by setting the [direction](#) property to **AntiClockWise**. For more information on axis, click [here](#).

The following example illustrates the right to left rendering of the Circular Gauge.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CircularGaugeModule } from '@syncfusion/ej2-angular-circulargauge'
import { LegendService, GaugeTooltipService } from '@syncfusion/ej2-angular-circulargauge'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    CircularGaugeModule
  ],
  providers: [LegendService, GaugeTooltipService],
  standalone: true,
  selector: 'app-container',
  template:
    `

```

```

        height: 10,
        offset: 5,
        color: '#9E9E9E'
    };
    this.minorTicks = {
        height: 0
    };
    this.labelStyle= {
        position: 'Inside', useRangeColor: false,
        font: {
            size: '12px',
            color: '#424242',
            fontFamily: 'Roboto',
            fontStyle: 'Regular'
        }
    };
    this.tooltip = {
        type: ['Pointer', 'Range'],
        format: 'Pointer : {value} ',
        enable: true,
        enableAnimation: false
    };
    this.legendSettings= {
        visible: true
    };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Ej1 api migration in Angular Circular gauge component

This article describes the API migration process of Accordion component from Essential JS 1 to Essential JS 2.

Circular gauge dimensions

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

|Height| **Property:** *height*
`<ej-circulargauge id="gauge" height="150"></ej-circulargauge>` | **Property:** *height*
`<ejs-circulargauge id="gauge" height="150px"></ejs-circulargauge>` |

|Width| **Property:** *width*
`<ejs-circulargauge id="gauge" width="200"></ej-circulargauge>` | **Property:** *width*
`<ej-circulargauge id="gauge" width="200px"></ej-circulargauge>` |

|Height(In Percentage)| Not Applicable | **Property:** *height*
`<ejs-circulargauge id="gauge" height='70%'></ejs-circulargauge>` |

|Width(In Percentage)| Not Applicable |**Property:** *width*

 <ejs-circulargauge id="gauge" width='100%'> </ejs-circulargauge>|

Axis Line

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Axisline Width| **Property:** *scales.size*

 <ej-circulargauge id="gauge"><e-scales><e-scale showScaleBar= "true", size= "6" ></e-scale></e-scales></ej-circulargauge>|**Property:** *axes.lineStyle.width*

 <ejs-circulargauge id="gauge"><e-axes><e-axis ></e-axis></e-axes></ejs-circulargauge>|

|Axisline Color| **Property:** *scales.size*

 <ej-circulargauge id="gauge"><e-scales><e-scale showScaleBar= "true" backgroundColor= "red" ></e-scale></e-scales></ej-circulargauge>|**Property:** *axes.lineStyle.width*

 <ejs-circulargauge id="gauge"><e-axes><e-axis ></e-axis></e-axes></ejs-circulargauge>|

|Axisline BackgroundColor| Not Applicable |**Property:** *axes.background*

 <ejs-circulargauge id="gauge"><e-axes><e-axis background='red'></e-axis></e-axes></ejs-circulargauge>|

|Axisline Direction| **Property:** *scales.direction*

 <ej-circulargauge id="gauge"><e-scales><e-scale direction= "counterclockwise"></e-scale></e-scales></ej-circulargauge>|**Property:** *axes.direction*

 <ejs-circulargauge id="gauge"><e-axes><e-axis direction='AntiClockWise'></e-axis></e-axes></ejs-circulargauge>|

|Axisline Radius| **Property:** *scales.radius*

 <ej-circulargauge id="gauge"><e-scales><e-scale showScaleBar= "true" radius= "150" ></e-scale></e-scales></ej-circulargauge>|**Property:** *axes.radius*

 <ejs-circulargauge id="gauge"><e-axes><e-axis radius='150'></e-axis></e-axes></ejs-circulargauge>|

|Axisline Startangle| **Property:** *scales.startAngle*

 <ej-circulargauge id="gauge"><e-scales><e-scale startAngle=80 ></e-scale></e-scales></ej-circulargauge>|**Property:** *axes.startAngle*

 <ejs-circulargauge id="gauge"><e-axes><e-axis startAngle=200></e-axis></e-axes></ejs-circulargauge>|

|Axisline Endangle| **Property:** *scales.sweepAngle*

 <ej-circulargauge id="gauge"><e-scales><e-scale sweepAngle= 250 ></e-scale></e-scales></ej-circulargauge>|**Property:** *axes.endAngle*

 <ejs-circulargauge id="gauge"><e-axes><e-axis endAngle= 150 ></e-axis></e-axes></ejs-circulargauge>|

|Minimum Axisvalue| **Property:** *scales.minimum*

 <ej-circulargauge id="gauge"><e-scales><e-scale minimum= 20 ></e-scale></e-scales></ej-circulargauge>|**Property:** *axes.minimum*

 <ejs-circulargauge id="gauge"><e-axes><e-axis minimum= 20></e-axis></e-axes></ejs-circulargauge>|

|Maximum Axisvalue| **Property:** *scales.maximum*

 <ej-circulargauge id="gauge"><e-scales><e-scale maximum= 200 ></e-scale></e-scales></ej-circulargauge>|**Property:** *axes.maximum*

 <ejs-circulargauge id="gauge"><e-axes><e-axis maximum= 200></e-axis></e-axes></ejs-circulargauge>|

Ticks

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Type of Ticks | **Property:** *scales.ticks.type*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-ticks><e-tick type="major"></e-tick></e-ticks></e-scale></e-scales> </ej-circulargauge>` | **Property:** *axes.majorTicks.height*
`<ejs-circulargauge id="gauge"><e-axes><e-axis [majorTicks]='majorTicks'></e-axis></e-axes> </ejs-circulargauge>`
 public majorTicks: Object= { }|

| Height of Major Ticks | **Property:** *scales.ticks.height*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-ticks><e-tick type="major" height= "8"></e-tick></e-ticks></e-scale></e-scales> </ej-circulargauge>` | **Property:** *axes.majorTicks.height*
`<ejs-circulargauge id="gauge"><e-axes><e-axis [majorTicks]='majorTicks'></e-axis></e-axes> </ejs-circulargauge>`
 public majorTicks: Object= { height: 8 }|

| Width of Major Ticks | **Property:** *scales.ticks.width*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-ticks><e-tick type="major" width= "5"></e-tick></e-ticks></e-scale></e-scales> </ej-circulargauge>` | **Property:** *axes.majorTicks.width*
`<ejs-circulargauge id="gauge"><e-axes><e-axis [majorTicks]='majorTicks'></e-axis></e-axes> </ejs-circulargauge>`
 public majorTicks: Object= { width: 5 }|

| Color of Major Ticks | **Property:** *scales.ticks.color*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-ticks><e-tick type="major" color='blue'></e-tick></e-ticks></e-scale></e-scales> </ej-circulargauge>` | **Property:** *axes.majorTicks.color*
`<ejs-circulargauge id="gauge"><e-axes><e-axis [majorTicks]='majorTicks'></e-axis></e-axes> </ejs-circulargauge>`
 public majorTicks: Object= { color:'blue' }|

| Offset for Major Ticks | **Property:** *scales.ticks.distanceFromScale*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-ticks><e-tick type="major" [distanceFromScale]="10"></e-tick></e-ticks></e-scale></e-scales> </ej-circulargauge>` | **Property:** *axes.majorTicks.offset*
`<ejs-circulargauge id="gauge"><e-axes><e-axis [majorTicks]='majorTicks'></e-axis></e-axes> </ejs-circulargauge>`
 public majorTicks: Object= { offset : 1 }|

| Interval of Major Ticks | **Property:** *scales.majorIntervalValue*
`<ej-circulargauge id="gauge"><e-scales><e-scale majorIntervalValue= "15"></e-scale></e-scales> </ej-circulargauge>` | **Property:** *axes.majorTicks.interval*
`<ejs-circulargauge id="gauge"><e-axes><e-axis [majorTicks]='majorTicks'></e-axis></e-axes> </ejs-circulargauge>`
 public majorTicks: Object= { interval : 15 }|

| Angle of Major Ticks | **Property:** *scales.ticks.angle*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-ticks><e-tick type="major" angle= "30"></e-tick></e-ticks></e-scale></e-scales> </ej-circulargauge>` | Not Applicable|

| Height of Minor Ticks | **Property:** *scales.ticks.height*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-ticks><e-tick type="minor" height= "8"></e-tick></e-ticks></e-scale></e-scales> </ej-circulargauge>` | **Property:** *axes.minorTicks.height*
`<ejs-circulargauge`

id="gauge"><e-axes><e-axis [minorTicks]='minorTicks'></e-axis></e-axes> </ejs-circulargauge>
</>public minorTicks: Object= { height: 8 }|

| Width of Minor Ticks| **Property:** *scales.ticks.width*

<ej-circulargauge id="gauge"><e-scales><e-scale><e-ticks><e-tick type="minor" width= "5"></e-tick></e-ticks></e-scale></e-scales> </ej-circulargauge>| **Property:** *axes.minorTicks.width*

<ejs-circulargauge id="gauge"><e-axes><e-axis [minorTicks]='minorTicks'></e-axis></e-axes> </ejs-circulargauge>
</>public minorTicks: Object= { width: 5 }|

| Color of Minor Ticks| **Property:** *scales.ticks.color*

<ej-circulargauge id="gauge"><e-scales><e-scale><e-ticks><e-tick type="minor" color='blue'></e-tick></e-ticks></e-scale></e-scales> </ej-circulargauge>| **Property:** *axes.minorTicks.color*

<ejs-circulargauge id="gauge"><e-axes><e-axis [minorTicks]='minorTicks'></e-axis></e-axes> </ejs-circulargauge>
</>public minorTicks: Object= { color:'blue' }|

| Offset for Major Ticks| **Property:** *scales.ticks.distanceFromScale*

<ej-circulargauge id="gauge"><e-scales><e-scale><e-ticks><e-tick type="minor" [distanceFromScale]="10"></e-tick></e-ticks></e-scale></e-scales></ej-circulargauge>| **Property:** *axes.minorTicks.offset*

<ejs-circulargauge id="gauge"><e-axes><e-axis [minorTicks]='minorTicks'></e-axis></e-axes> </ejs-circulargauge>
</>public minorTicks: Object= { offset : 1 }|

| Interval of Minor Ticks| **Property:** *scales.majorIntervalValue*

<ej-circulargauge id="gauge"><e-scales><e-scale minorIntervalValue= "15"></e-scale></e-scales> </ej-circulargauge>| **Property:** *axes.minorTicks.interval*

<ejs-circulargauge id="gauge"><e-axes><e-axis [minorTicks]='minorTicks'></e-axis></e-axes> </ejs-circulargauge>
</>public minorTicks: Object= { interval : 15 }|

| Angle of Minor Ticks| **Property:** *scales.ticks.angle*

<ej-circulargauge id="gauge"><e-scales><e-scale><e-ticks><e-tick type="minor" angle= "30"></e-tick></e-ticks></e-scale></e-scales> </ej-circulargauge>| Not Applicable|

Labels

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Autoangle| **Property:** *scales.labels.autoAngle*

<ej-circulargauge id="gauge"><e-scales><e-scale [labels]="labels"></e-scale></e-scales> </ej-circulargauge>
</>public labels:Object[] = [{ showLabels: true,autoAngle: true}]| **Property:** *axes.labelStyle.autoAngle*

<ejs-circulargauge id="gauge"><e-axes><e-axis [labelStyle]="labelStyle"></e-axis></e-axes> </ejs-circulargauge>
</>public labelStyle: Object= { autoAngle: true }|

| Angle| **Property:** *scales.labels.angle*

<ej-circulargauge id="gauge"><e-scales><e-scale [labels]="labels"></e-scale></e-scales> </ej-circulargauge>
</>public labels:Object[] = [{ showLabels: true,angle: 10}]| Not Applicable|

| Offset| **Property:** *scales.labels.distanceFromScale*

<ej-circulargauge id="gauge"><e-scales><e-scale [labels]="labels"></e-scale></e-scales> </ej-

circulargauge>

publiclabels:Object[] = [{ showLabels: true,distanceFromScale: { x: 0, y: 60 } }]| **Property:** axes.labelStyle.offset

 <ejs-circulargauge id="gauge"><e-axes><e-axis [labelStyle]= "labelStyle"></e-axis></e-axes> </ejs-circulargauge>

public labelStyle: Object= { offset : 3 }|

|Format| **Property:** scales.labels.unitText

 <ej-circulargauge id="gauge"><e-scales><e-scale [labels]="labels"></e-scale></e-scales> </ej-circulargauge>

publiclabels:Object[] = [{ unitText: "F"}]| **Property:** axes.labelStyle.format

 <ejs-circulargauge id="gauge"><e-axes><e-axis [labelStyle]= "labelStyle"></e-axis></e-axes> </ejs-circulargauge>

public labelStyle: Object= { format: 'C' }|

|Unit Text Placement| **Property:** scales.labels.unitTextPlacement

<ej-circulargauge id="gauge"><e-scales><e-scale [labels]="labels"></e-scale></e-scales> </ej-circulargauge>

publiclabels:Object[] = [{ showLabels: true,unitTextPlacement: "front"}]| Not Applicable|

|Label Range Color| Not Applicable| **Property:** axes.labelStyle.useRangeColor

 <ejs-circulargauge id="gauge"><e-axes><e-axis [labelStyle]= "labelStyle"></e-axis></e-axes> </ejs-circulargauge>

public labelStyle: Object= { useRangeColor: true }|

|Opacity| **Property:** scales.labels.opacity

 <ej-circulargauge id="gauge"><e-scales><e-scale [labels]="labels"></e-scale></e-scales> </ej-circulargauge>

publiclabels:Object[] = [{opacity: 0.5}]| **Property:** axes.labelStyle.font.opacity

 <ejs-circulargauge id="gauge"><e-axes><e-axis [labelStyle]= "labelStyle"></e-axis></e-axes> </ejs-circulargauge>

public labelStyle: Object= { font: { opacity: 5 } }|

|Label Text Color| **Property:** scales.labels.textColor

 <ej-circulargauge id="gauge"><e-scales><e-scale [labels]="labels"></e-scale></e-scales> </ej-circulargauge>

publiclabels:Object[] = [{ textColor: "Red",}]| **Property:** axes.labelStyle.font.color

 <ejs-circulargauge id="gauge"><e-axes><e-axis [labelStyle]= "labelStyle"></e-axis></e-axes> </ejs-circulargauge>

public labelStyle: Object= { font:{ color: 'red' } }|

|Label Font Family| **Property:** scales.labels.font.fontFamily

<ej-circulargauge id="gauge"><e-scales><e-scale [labels]="labels"></e-scale></e-scales> </ej-circulargauge>

publiclabels:Object[] = [{font:{ fontFamily: 'Arial' } }]| **Property:** axes.labelStyle.font.fontFamily

 <ejs-circulargauge id="gauge"><e-axes><e-axis [labelStyle]= "labelStyle"></e-axis></e-axes> </ejs-circulargauge>

public labelStyle: Object= { font:{ fontFamily: 'Arial' } }|

|Label Font Style| **Property:** scales.labels.font.fontStyle

 <ej-circulargauge id="gauge"><e-scales><e-scale [labels]="labels"></e-scale></e-scales> </ej-circulargauge>

publiclabels:Object[] = [{ font: { fontStyle: 'Bold' } }]| **Property:** axes.labelStyle.font.fontStyle

 <ejs-circulargauge id="gauge"><e-axes><e-axis [labelStyle]= "labelStyle"></e-axis></e-axes> </ejs-circulargauge>

public labelStyle: Object= { font: { fontStyle: 'Bold' } }|

| Label Size | **Property:** *scales.labels.font.size*
`<ej-circulargauge id="gauge"><e-scales><e-scale [labels]="labels"></e-scale></e-scales></ej-circulargauge>

publicLabels:Object[] = [{font:{ size: "15px" } }]| Property: axes.labelStyle.font.size
<ejs-circulargauge id="gauge"><e-axes><e-axis [labelStyle]= "labelStyle"></e-axis></e-axes></ejs-circulargauge>

public labelStyle: Object= { font:{ size: "15px" } }|`

| Label Font Weight | Not Applicable | **Property:** *axes.labelStyle.font.fontWeight*
`<ejs-circulargauge id="gauge"><e-axes><e-axis [labelStyle]= "labelStyle"></e-axis></e-axes></ejs-circulargauge>

public labelStyle: Object= { font:{ fontWeight: '4' } }|`

Ranges

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Start Value | **Property:** *scales.ranges.startValue*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-ranges><e-range showRanges="true" startValue="20"></e-range></e-ranges></e-scale></e-scales></ej-circulargauge>| Property: axes.ranges.start
<ejs-circulargauge id="gauge"><e-axes><e-axis ><e-ranges><e-range start= "20" ></e-range></e-ranges></e-axis></e-axes></ejs-circulargauge>|`

| End Value | **Property:** *scales.ranges.endValue*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-ranges><e-range showRanges="true" endValue= "20" ></e-range></e-ranges></e-scale></e-scales></ej-circulargauge>| Property: axes.ranges.end
<ejs-circulargauge id="gauge"><e-axes><e-axis ><e-ranges><e-range end= "20"></e-range></e-ranges></e-axis></e-axes></ejs-circulargauge>|`

| Start Width | **Property:** *scales.ranges.startWidth*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-ranges><e-range showRanges="true" startWidth= "10" ></e-range></e-ranges></e-scale></e-scales></ej-circulargauge>| Property: axes.ranges.startWidth
<ejs-circulargauge id="gauge"><e-axes><e-axis ><e-ranges><e-range startWidth= "10"></e-range></e-ranges></e-axis></e-axes></ejs-circulargauge>|`

| End Width | **Property:** *scales.ranges.endWidth*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-ranges><e-range showRanges="true" endWidth= "15"></e-range></e-ranges></e-scale></e-scales></ej-circulargauge>| Property: axes.ranges.endWidth
<ejs-circulargauge id="gauge"><e-axes><e-axis ><e-ranges><e-range endWidth= "15"></e-range></e-ranges></e-axis></e-axes></ejs-circulargauge>|`

| Color | **Property:** *scales.ranges.backgroundColor*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-ranges><e-range showRanges="true" backgroundColor= "red" ></e-range></e-ranges></e-scale></e-scales></ej-circulargauge>| Property: axes.ranges.color
<ejs-circulargauge id="gauge"><e-axes><e-axis ><e-ranges><e-range color= "red"></e-range></e-ranges></e-axis></e-axes></ejs-circulargauge>|`

| Offset | **Property:** *scales.ranges.distanceFromScale*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-ranges><e-range showRanges="true" distanceFromScale= "10" ></e-range></e-ranges></e-scale></e-scales></ej-circulargauge>| Property:`

axes.ranges.offset
`<ejs-circulargauge id="gauge"><e-axes><e-axis ><e-ranges><e-range offset= "10" ></e-range></e-ranges></e-axis></e-axes> </ejs-circulargauge>|`

| Range Position | **Property:** *scales.ranges.placement*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-ranges><e-range showRanges="true" placement="Near"></e-range></e-ranges></e-scale> </e-scales> </ej-circulargauge>| Property: axes.ranges.position
<ejs-circulargauge id="gauge"><e-axes><e-axis ><e-ranges><e-range position= 'Inside' ></e-range></e-ranges></e-axis></e-axes> </ejs-circulargauge>|`

| Opacity | **Property:** *scales.ranges.opacity*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-ranges><e-range showRanges="true" opacity= "0.3"></e-range></e-ranges> </e-scale> </e-scales> </ej-circulargauge>| Not Applicable |`

| Radius | Not Applicable | **Property:** *axes.ranges.radius*
`<ejs-circulargauge id="gauge"><e-axes><e-axis ><e-ranges><e-range radius= '80' ></e-range></e-ranges></e-axis></e-axes> </ejs-circulargauge>|`

| Rounded Corner Radius | Not Applicable | **Property:** *axes.ranges.roundedCornerRadius*
`<ejs-circulargauge id="gauge"><e-axes><e-axis ><e-ranges><e-range roundedCornerRadius= 10 ></e-range></e-ranges></e-axis></e-axes> </ejs-circulargauge>|`

| Gradients | **Property:** *scales.ranges.gradients*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-ranges><e-range showRanges="true" [gradients] = " gradients" > </e-range></e-ranges> </e-scale> </e-scales> </ej-circulargauge>

public gradients:Object= { colorInfo: [{ colorStop : 0, color:"#FFFFFF" }] }| Not Applicable |`

| Border | **Property:** *scales.ranges.border*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-ranges><e-range showRanges="true" [border]= " border" > </e-range></e-ranges> </e-scale> </e-scales> </ej-circulargauge>

public border:Object= { color: "blue", width: 2 }| Not Applicable |`

Needle Pointer

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Needle Pointer | **Property:** *scales.pointers.type*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-pointers><e-pointer type= 'needle'><e-pointer></e-pointers></e-scale> </e-scales></ej-circulargauge>| Property: axes.pointers.type
<ejs-circulargauge id="gauge" type= 'needle' value= 20 ><e-axes> <e-axis><e-pointers><e-pointer ></e-pointer></e-pointers></e-axis></e-axes></ejs-circulargauge>|`

| Needle Pointer Color | **Property:** *scales.pointers.backgroundColor*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-pointers><e-pointer type='needle' backgroundColor='red'><e-pointer></e-pointers></e-scale></e-scales></ej-circulargauge>| Property: axes.pointers.color
<ejs-circulargauge id="gauge"><e-axes><e-axis><e-pointers><e-pointer type= 'needle' color: 'red'></e-pointer></e-pointers></e-axis></e-axes></ejs-circulargauge>|`

| Animation | **Property:** *enableAnimation*
`<ej-circulargauge id="gauge" enableAnimation="true" ></ej-circulargauge>` | **Property:** *axes.pointers.animation*
`<ej-circulargauge id="gauge"><e-axes><e-axis><e-pointers><e-pointer animation="true" duration=1000 ></e-pointer></e-pointers></e-axis></e-axes></ej-circulargauge>` |

| Pointer Width | **Property:** *scales.pointers.width*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-pointers><e-pointer width= 5 ></e-pointer></e-pointers></e-scale> </e-scales> </ej-circulargauge>` | **Property:** *axes.pointers.pointerWidth*
`<ej-circulargauge id="gauge"><e-axes><e-axis><e-pointers><e-pointer pointerWidth=5></e-pointer></e-pointers></e-axis></e-axes></ej-circulargauge>` |

| Pointer Radius | **Property:** *scales.pointers.distanceFromScale*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-pointers><e-pointer distanceFromScale=10></e-pointer></e-pointers></e-scale></e-scales></ej-circulargauge>` | **Property:** *axes.pointers.radius*
`<ej-circulargauge id="gauge"><e-axes><e-axis><e-pointers><e-pointer radius: 80></e-pointer></e-pointers></e-axis></e-axes></ej-circulargauge>` |

| Opacity | **Property:** *scales.pointers.opacity*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-pointers><e-pointer opacity=0.5></e-pointer></e-pointers></e-scale></e-scales> </ej-circulargauge>` | Not Applicable |

| Needle Type | **Property:** *scales.pointers.needleType*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-pointers><e-pointer type= 'needle' needleType="triangle"></e-pointer> </e-pointers></e-scale></e-scales></ej-circulargauge>` | Not Applicable |

| Back Needle Length | **Property:** *scales.pointers.backNeedleLength*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-pointers><e-pointer showBackNeedle= "true" backNeedleLength= 3 ></e-pointer></e-pointers></e-scale></e-scales></ej-circulargauge>` | **Property:** *axes.pointers.needleTail.length*
`<ej-circulargauge id="gauge"><e-axes><e-axis><e-pointers><e-pointer [needleTail] ="needleTail"></e-pointer></e-pointers> </e-axis></e-axes></ej-circulargauge>`
 public needleTail: Object = { length: 5 } |

Marker Pointer

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Marker Pointer | **Property:** *scales.pointers.type*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-pointers><e-pointer type='marker'></e-pointer></e-pointers></e-scale></e-scales></ej-circulargauge>` | **Property:** *axes.pointers.type*
`<ej-circulargauge id="gauge"><e-axes><e-axis><e-pointers><e-pointer type= 'marker' value= 20></e-pointer></e-pointers> </e-axis></e-axes></ej-circulargauge>` |

| Marker Type | **Property:** *scales.pointers.markerType*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-pointers><e-pointer type= 'marker' markerType= "rectangle"></e-pointer> </e-pointers></e-scale></e-scales></ej-circulargauge>` | **Property:** *axes.pointers.markerShape*
`<ej-circulargauge id="gauge"><e-axes><e-axis><e-pointers><e-pointer type= 'marker' markerShape= 'Diamond' ></e-pointer></e-pointers></e-axis></e-axes></ej-circulargauge>` |

| Marker Width | **Property:** *scales.pointers.width*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-pointers><e-pointer type= 'marker' width= 20><e-pointer></e-pointers></e-scale></e-scales></ej-circulargauge>` | **Property:** *axes.pointers.markerWidth*
`<ej-circulargauge id="gauge"><e-axes><e-axis><e-pointers><e-pointer type= 'marker' markerWidth= 20 ></e-pointer></e-pointers></e-axis></e-axes></ej-circulargauge>` |

| Marker Height | **Property:** *scales.pointers.length*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-pointers><e-pointer type= 'marker' length= 25><e-pointer></e-pointers></e-scale></e-scales></ej-circulargauge>` | **Property:** *axes.pointers.markerHeight*
`<ej-circulargauge id="gauge"><e-axes><e-axis><e-pointers><e-pointer type= 'marker' markerHeight= 25> </e-pointer></e-pointers></e-axis></e-axes></ej-circulargauge>` |

| Marker Image | **Property:** *scales.pointers.imageUrl*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-pointers><e-pointer type= 'marker' imageUrl= "football.png"><e-pointer></e-pointers></e-scale></e-scales></ej-circulargauge>` | **Property:** *axes.pointers.imageUrl*
`<ej-circulargauge id="gauge"><e-axes><e-axis><e-pointers><e-pointer type= 'marker' imageUrl= "football.png"></e-pointer></e-pointers></e-axis></e-axes></ej-circulargauge>` |

| Border Customization | **Property:** *scales.pointers.border*
`<ej-circulargauge id="gauge"><e-scales><e-scale><e-pointers><e-pointer type= 'marker' [border]="border"><e-pointer> </e-pointers></e-scale></e-scales></ej-circulargauge>`
 public border: Object= { color: 'red', width: 2 } | **Property:** *axes.pointers.border*
`<ej-circulargauge id="gauge"><e-axes><e-axis><e-pointers><e-pointer type= 'marker' [border]="border"></e-pointer> </e-pointers></e-axis></e-axes></ej-circulargauge>`
 public border: Object= { color: 'red', width: 2 } |

Rangebar Pointer

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Rangebar | Not Applicable | **Property:** *axes.pointers.type*
`<ej-circulargauge id="gauge"><e-axes><e-axis><e-pointers><e-pointer type: 'RangeBar'></e-pointer></e-pointers> </e-axis></e-axes></ej-circulargauge>` |

| Rounded Corner Radius | Not Applicable | **Property:** *axes.pointers.roundedCornerRadius*
`<ej-circulargauge id="gauge"><e-axes><e-axis><e-pointers><e-pointer type= 'RangeBar' roundedCornerRadius= 10 ></e-pointer></e-pointers> </e-axis></e-axes></ej-circulargauge>` |

Annotations

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Content | **Property:** *scales.customLabels.value*
`<ej-circulargauge id="gauge"><e-scales><e-scale [customLabels]="customLabels"></e-scale></e-scales></ej-circulargauge>`
 public customLabels:Object = {value: 'Lineargauge' } | **Property:** *axes.annotations.content*
`<ej-circulargauge id="gauge"><e-axes><e-axis><e-`


```

annotations><e-annotation content='Annotation'></e-annotation> </e-annotations> </e-
axis></e-axes> </ejs-circulargauge>|

```

```

|Angle| Property: scales.customLabels.textAngle<br/><br/> <ej-circulargauge id="gauge"><e-
scales><e-scale [customLabels]="customLabels"></e-scale></e-scales></ej-
circulargauge><br/><br/>public customLabels:Object = {textAngle: 90}| Property:
axes.annotations.angle<br/><br/> <ej-circulargauge id="gauge"><e-axes><e-axis><e-
annotations><e-annotation angle= 90 ></e-annotation> </e-annotations> </e-axis></e-axes>
</ejs-circulargauge>|

```

```

|Font Family| Property: scales.customLabels.font.fontFamily<br/><br/> <ej-circulargauge
id="gauge"><e-scales><e-scale [customLabels]="customLabels"></e-scale></e-scales></ej-
circulargauge><br/><br/>public customLabels:Object = {font: { fontFamily: "Arial" }}| Property:
axes.annotations.textStyle.fontFamily<br/><br/> <ej-circulargauge id="gauge"><e-axes><e-
axis><e-annotations><e-annotation [ textStyle] ="textStyle"></e-annotation> </e-
annotations></e-axis></e-axes> </ejs-circulargauge><br/><br/>public textStyle: Object=
{fontFamily: "Arial" }|

```

```

|Font Color| Property: scales.customLabels.color<br/><br/> <ej-circulargauge id="gauge"><e-
scales><e-scale [customLabels]="customLabels"></e-scale> </e-scales></ej-
circulargauge><br/><br/>public customLabels:Object = {color : "red"}| Property:
axes.annotations.textStyle.color<br/><br/> <ej-circulargauge id="gauge"><e-axes><e-axis><e-
annotations><e-annotation [ textStyle] ="textStyle"></e-annotation> </e-annotations></e-
axis></e-axes> </ejs-circulargauge><br/><br/>public textStyle: Object= {color: "red" }|

```

```

|Auto Angle| Not Applicable| Property: axes.annotations.autoAngle<br/><br/> <ej-circulargauge
id="gauge"><e-axes><e-axis><e-annotations><e-annotation autoAngle = true></e-
annotation></e-annotations> </e-axis> </e-axes> </ejs-circulargauge>|

```

```

|Radius| Not Applicable| Property: axes.annotations.radius<br/><br/> <ej-circulargauge
id="gauge"><e-axes><e-axis><e-annotations><e-annotation radius = "10%"></e-
annotation></e-annotations></e-axis> </e-axes> </ejs-circulargauge>|

```

```

|Annotation Position| Property: scales.customLabels.position<br/><br/> <ej-circulargauge
id="gauge"><e-scales><e-scale [customLabels]="customLabels"></e-scale></e-scales></ej-
circulargauge><br/><br/>public customLabels:Object = {position : { x: 10, y: 10 }}| Not Applicable|

```

```

|Annotation Position Type| Property: scales.customLabels.positionType<br/><br/> <ej-circulargauge
id="gauge"><e-scales><e-scale [customLabels]="customLabels"></e-scale></e-scales></ej-
circulargauge><br/><br/>public customLabels:Object = {positionType : "outer"}| Not Applicable|

```

```

|ZIndex| Not Applicable| Property: axes.annotations.zIndex<br/><br/> <ej-circulargauge
id="gauge"><e-axes><e-axis><e-annotations><e-annotation zIndex = '1' ></e-annotation></e-
annotations> </e-axis></e-axes> </ejs-circulargauge>|

```

Appearance

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

|Title| Not Applicable| **Property:** *title*

 <ejs-circulargauge id="gauge" title= 'Circular Gauge' ></ejs-circulargauge>|

|Background Color| **Property:** *backgroundColor*

 <ej-circulargauge id="gauge" backgroundColor= "red"></ej-circulargauge>| **Property:** *background*

 <ejs-circulargauge id="gauge" background="red" ></ejs-circulargauge>|

|Localization| **Property:** *locale*

 \$<ej-circulargauge id="gauge" locale = "en-US" ></ej-circulargauge>| **Property:** *locale*

 <ejs-circulargauge id="gauge" locale = "en-US" ></ejs-circulargauge>;|

|Border| Not Applicable| **Property:** *border*

 <ejs-circulargauge id="gauge" [border] = "border"></ejs-circulargauge>
</br>public border: Object= {color: "red" , width: 2}|

|Center of X| Not Applicable| **Property:** *centerX*

 <ejs-circulargauge id="gauge" centerX = "120px"></ejs-circulargauge>|

|Center of Y| Not Applicable| **Property:** *centerY*

 <ejs-circulargauge id="gauge" centerY = "150px" ></ejs-circulargauge>|

|Theme| **Property:** *theme*

 <ej-circulargauge id="gauge" theme = "flatlight" ></ej-circulargauge>| **Property:** *theme*

 <ejs-circulargauge id="gauge" theme = "Material" ></ejs-circulargauge>|

|Margin| Not Applicable| **Property:** *margin*

 <ejs-circulargauge id="gauge" [margin] = "margin"></ejs-circulargauge>
</br>public margin: Object= {left: 40, right: 40, top: 40, bottom: 40}|

Events

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Annotation Event| **Event:** *drawCustomLabel*

 <ej-circulargauge id="gauge" (drawCustomLabel)="onDrawCustomLabel(\$event)"></ej-circulargauge>

onDrawCustomLabel(sender){}| **Event:** *annotationRender*

 <ejs-circulargauge id="gauge" (annotationRender)= 'annotationRender(\$event)'></ejs-circulargauge>

public annotationRender(args: IAnnotationRenderEventArgs): void {}|

|Label Event| **Event:** *drawLabels*

 <ej-circulargauge id="gauge" (drawLabels)="DrawLabels(\$event)"></ej-circulargauge>

DrawLabels(sender){}| **Event:** *axisLabelRender*

 <ejs-circulargauge id="gauge" (axisLabelRender)= 'axisLabelRender(\$event)'></ejs-circulargauge>

public axisLabelRender(args: IAxisLabelRenderEventArgs): void {}|

|Load Event| **Event:** *load*

 <ej-circulargauge id="gauge" (load)="onLoad(\$event)"></ej-circulargauge>

onLoad(sender){}| **Event:** *load*

 <ejs-circulargauge id="gauge" (load)= 'load(\$event)'></ejs-circulargauge>

public load(args: ILoadedEventArgs): void {}|

|Loaded Event| **Event:** *loaded*

 <ej-circulargauge id="gauge" (loaded)="onLoaded(\$event)"></ej-circulargauge>

onLoaded(sender){}| **Event:**

loaded

 <ejs-circulargauge id="gauge" (loaded)=*'loaded(\$event)'*></ejs-circulargauge>

public loaded(args: ILoadedEventArgs): void {}|

| Tooltip Rendered Event| Not Applicable| **Event:** *tooltipRender*

 <ejs-circulargauge id="gauge" (tooltipRender)=*'tooltipRender(\$event)'*></ejs-circulargauge>

public tooltipRender(args: ITooltipRenderEventArgs): void {}|

| Resized Rendered Event| Not Applicable| **Event:** *resized*

<ejs-circulargauge id="gauge" (tooltipRender)=*'tooltipRender(\$event)'*></ejs-circulargauge>

public tooltipRender(args: IResizeEventArgs): void {}|

| Animation Event| Not Applicable| **Event:** *animationComplete*

 <ejs-circulargauge id="gauge" (animationComplete)=*'animationComplete(\$event)'*></ejs-circulargauge>

public animationComplete(args: IAnimationCompleteEventArgs): void {}|

| Mousedown Event| **Event:** *mouseClick*

 \$<ej-circulargauge id="gauge" (mouseClick)=*"onMouseClicked(\$event)"*></ej-circulargauge>

onMouseClicked(sender){}| **Event:** *gaugeMouseDown*

 <ejs-circulargauge id="gauge" (gaugeMouseDown)=*'gaugeMouseDown(\$event)'*></ejs-circulargauge>

public gaugeMouseDown(args: IMouseEventArgs): void {}|

| Mousemove Event| **Event:** *mouseClickMove*

 <ej-circulargauge id="gauge" (mouseClickMove)=*"onMouseClickedMove(\$event)"*></ej-circulargauge>

onMouseClickedMove(sender){}| **Event:** *gaugeMouseLeave*

 <ejs-circulargauge id="gauge" (gaugeMouseLeave)=*'gaugeMouseLeave(\$event)'*></ejs-circulargauge>

public gaugeMouseLeave(args: IMouseEventArgs): void {}|

| Mouseup Event| **Event:** *mouseClickUp*

 \$<ej-circulargauge id="gauge" (mouseClickUp)=*"onMouseClickedUp(\$event)"*></ej-circulargauge>

onMouseClickedUp(sender){}| **Event:** *gaugeMouseUp*

 <ejs-circulargauge id="gauge" (gaugeMouseUp)=*'gaugeMouseUp(\$event)'*></ejs-circulargauge>

public gaugeMouseUp(args: IMouseEventArgs): void {}|

| Pointerdrag Move Event| **Event:** *drawPointers*

 <ej-circulargauge id="gauge" (drawPointers)=*"ondrawPointers(\$event)"*></ej-circulargauge>

onDrawPointers(sender){}| **Event:** *dragMove*

 <ejs-circulargauge id="gauge" (dragMove)=*'dragMove(\$event)'*></ejs-circulargauge>

public dragMove(args: IMouseEventArgs): void {}|

| Draw Range Event| **Event:** *drawRange*

 \$<ej-circulargauge id="gauge" (drawRange)=*"onDrawRange(\$event)"*></ej-circulargauge>

onDrawRange(sender){}| Not Applicable|

| Draw Ticks Event| **Event:** *drawTicks*

 \$<ej-circulargauge id="gauge" (drawTicks)=*"onDrawTicks(\$event)"*></ej-circulargauge>

onDrawTicks(sender){}| Not Applicable|

| Legend Render Event| **Event:** *legendItemRender*

 <ej-circulargauge id="gauge" (legendItemRender)="onLegendItemRender(\$event)"></ej-circulargauge>

onLegendItemRender(sender){}| Not Applicable|

| Animation Complete Event| Not Applicable| **Event:** *animationComplete*

 <ejs-circulargauge id="gauge" (animationComplete)='animationComplete(\$event)'></ejs-circulargauge>

public animationComplete(args: IAnimationCompleteEventArgs): void {}|

| Right Click Event| **Event:** *rightClick*

 \$<ej-circulargauge id="gauge" (rightClick)="onRightClick(\$event)"></ej-circulargauge>

onRightClick(sender){}| Not Applicable|

| Double Click Event| **Event:** *doubleClick*

 \$<ej-circulargauge id="gauge" (doubleClick)="onDoubleClick(\$event)"></ej-circulargauge>

onDoubleClick(sender){}| Not Applicable|

ColorPicker

Getting started with Angular Color picker component

This section explains how to create a default ColorPicker and demonstrate the basic usage of the ColorPicker module.

Dependencies

The list of dependencies required to use the ColorPicker module in your application is given below:

```
`javascript
```

```
|-- @syncfusion/ej2-angular-inputs
```

```
|-- @syncfusion/ej2-angular-base
```

```
|-- @syncfusion/ej2-base
```

```
|-- @syncfusion/ej2-inputs
```

```
|-- @syncfusion/ej2-buttons
```

```
|-- @syncfusion/ej2-popups
```

```
|-- @syncfusion/ej2-splitbuttons
```

```
,
```

Setup Angular environment

You can use [Angular CLI](#) to setup your Angular applications. To install Angular CLI use the following command.

```
,
```

```
npm install -g @angular/cli
```

```
,
```

Create an Angular application

Start a new Angular application using below Angular CLI command.

```
`  
ng new my-app  
cd my-app  
`
```

Installing Syncfusion ColorPicker package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(`>=20.2.36`) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-inputs](#) package to the application.

```
`bash  
npm install @syncfusion/ej2-angular-inputs --save  
`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-inputs@ngcc](#) package to the application.

```
`bash  
npm install @syncfusion/ej2-angular-inputs@ngcc --save  
`
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash  
@syncfusion/ej2-angular-inputs:"20.2.38-ngcc"  
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding ColorPicker module

Import ColorPicker module into Angular application(`app.module.ts`) from the package

`@syncfusion/ej2-angular-inputs`.

```
`typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Importing Colorpicker module from Syncfusion ej2-angular-inputs package.
import { ColorPickerModule } from '@syncfusion/ej2-angular-inputs';
import { AppComponent } from './app.component';
@NgModule({
  imports: [ BrowserModule, ColorPickerModule ], // Declaration of ColorPickerModule into NgModule.
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
`
```

Adding Syncfusion ColorPicker component

Modify the template in `app.component.ts` file to render the ColorPicker component.

```
`typescript
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<!-- To render ColorPicker. -->
<input ej2-colorpicker type="color" id="colorpicker" />`
})
export class AppComponent { }
`
```

Adding CSS reference

Add ColorPicker component's styles as given below in `style.css`.

```
`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
`
```

Running the application

Run the application in the browser using the following command:

`

ng serve

`

The following example shows a default ColorPicker component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, ColorPickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <h4>Choose Color</h4>
    <!-- To render ColorPicker. -->
    <ejs-input ej2-colorpicker type="color" id="colorpicker"
  /></div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Inline type

By default, the ColorPicker will be rendered using SplitButton and open the pop-up to access the ColorPicker. To render the ColorPicker container alone and to access it directly, render it as inline. It can be achieved by setting the [inline](#) property to `true`.

The following sample shows the inline type rendering of ColorPicker.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
```

```

        FormsModule, ColorPickerModule
    ],
    standalone: true,
    selector: 'app-root',
    template: `<div class="e-section-control">
        <h4>Choose Color</h4>
        <!-- To render inline ColorPicker. -->
        <ejs-input ejs-colorpicker type="color" id="colorpicker"
[inline]="true" [showButtons]="false" /></div>`
    })
    export class AppComponent { }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

> The `showButtons` property is disabled in this sample because the control buttons are not needed for inline type. To know about the control buttons functionality, refer to the [showButtons](#) sample.

Note: You can refer to our [Angular color picker](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular ColorPicker example](#) that shows how to render the ColorPicker in Angular.

Mode and value in Angular Color picker component

Rendering palette at initial load

By default, the `Picker` area will be rendered at initial load. To render the `Palette` area while opening the ColorPicker pop-up, and specify the `mode` property as `Palette`.

In the following sample, it will render the `Palette` at initial load.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
    imports: [
        FormsModule, ColorPickerModule
    ],
    standalone: true,
    selector: 'app-root',
    template: `<div class="e-section-control">
        <h4>Select Color</h4>
        <ejs-input ejs-colorpicker type="color" id="element"
mode="Palette" /></div>`
    })
    export class AppComponent { }

```


MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Color value

The [value](#) property can be used to specify the color value to the ColorPicker. It supports either **three** or **six** digit hex codes. To include **opacity**, set the color value as **four** or **eight** digit hex code.

In the following sample, the color value sets as **four** digit hex code, the last digit represents the **opacity** value.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerComponent } from '@syncfusion/ej2-angular-inputs'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <h4>Choose Color</h4>
    <!-- To set color value. -->
    <ejs-input ej2-colorpicker type="color" id="element"
value="035a" /></div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

> The [value](#) property supports hex code with or without **#** prefix.

See Also

- [How to render palette alone](#)
- [Custom palette](#)
- [No color support in palette](#)

Localization in Angular Color picker component

Localization

The **Localization** library allows you to localize default text content of the ColorPicker. The ColorPicker component has static text for **control buttons (apply / cancel)** and **mode switcher** that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

The following list of properties and its values are used in the ColorPicker.

Locale key words | Text

Apply | Apply

Cancel | Cancel

ModeSwitcher | Switch Mode

Loading translations

To load translation object in an application use **load** function of **L10n** class.

The below example demonstrates the ColorPicker in **Deutsch** culture.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
import { L10n } from '@syncfusion/ej2-base';
L10n.load({
  'de-DE': {
    'colorpicker': {
      'Apply': 'Anwenden',
      'Cancel': 'Abbrechen',
      'ModeSwitcher': 'Modus wechseln'
    }
  }
});
@Component({
  imports: [
    FormsModule, ColorPickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <h4>Choose Color</h4>
    <ejs-input ej2-colorpicker type="color" id="element"
  locale="de-DE" /></div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Right to Left - RTL

ColorPicker component has **RTL** support. It helps to render the ColorPicker from right-to-left direction.

It improves the user experiences and accessibility for users who use right-to-left languages(Arabic, Farsi, Urdu, etc). This can be achieved by setting the [enableRtl](#) property to **true**.

The following example illustrates how to enable right-to-left support in ColorPicker component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
import { L10n } from '@syncfusion/ej2-base';
L10n.load({
  'ar-AE': {
    'colorpicker': {
      'Apply': 'تطبيق',
      'Cancel': 'إلغاء',
      'ModeSwitcher': 'مفتاح كهربائي الوضع'
    }
  }
});
@Component({
  imports: [
    FormsModule, ColorPickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <h4>Choose Color</h4>
    <ejs-input ej2-colorpicker type="color" id="element"
[enableRtl]="true" locale="ar-AE" /></div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [More information about localization](#)

Accessibility in Angular Color picker component

The Color picker component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Color picker component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Color picker component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Color picker component:

| Attributes | Purpose |

| --- | --- |

| **role** | Indicates the Color picker component as **color** and the tiles as **gridcell** in the color palette. |

| **aria-label** | Indicates the accessible name for the tiles. |

| **aria-selected** | Indicates the current selected state of the tile. |

| **aria-haspopup** | Indicates the availability of the popup element. |

| **aria-expanded** | Indicates whether the popup can be expanded or collapsed, as well as indicates whether its current state is expanded or collapsed. |

| **aria-owns** | Identifies an elements in order to define a visual, functional, or contextual parent/child relationship between DOM elements where the DOM hierarchy cannot be used to represent the relationship. |

| **aria-disabled** | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

Keyboard interaction

The Color picker component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Color picker component.

| Press | To do this |

| --- | --- |

| Up Arrow | Moves the handler/tile up from the current position. |

| Down Arrow | Moves the handler/tile down from the current position. |

| Left Arrow | Moves the handler/tile left from the current position. |

| Right Arrow | Moves the handler/tile right from the current position. |

| Enter | Apply the selected color value. |

| Tab | To focus the next focusable element in the Color picker popup. |

Ensuring accessibility

The Color picker component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Color picker component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Color picker component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Style and appearance in Angular Color picker component

To modify the ColorPicker appearance, you need to override the default CSS of ColorPicker component. Please find the list of CSS classes and its corresponding section in ColorPicker component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

| CSS Class | Purpose of Class |

|-----|-----|

|.e-custom-picker .e-container .e-handler|To customized Color Picker selection handler|

|.color-picker.e-dropdown-popup ul .e-container|To customize the Color Picker container|

|.color-picker.e-dropdown-popup ul .e-item.e-palette-item|To customize the Color Picker palette item|

|.color-picker.e-dropdown-popup .e-container .e-switch|To customize the Color Picker switch control|

|.color-picker.e-dropdown-popup .e-container .e-slider-preview|To customize the Color Picker slider control|

How To

Hide control buttons in Angular Color picker component

ColorPicker can be rendered without control buttons (Apply/Cancel). In this case, while selecting a color, the ColorPicker pop-up is closed and selected colors will be applied directly. To hide control buttons, set the [showButtons](#) property to `false`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, ColorPickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <h4>Choose Color</h4>
    <!-- To hide control buttons. -->
    <ejs-input ej2-colorpicker type="color" id="element"
[showButtons]="false" /></div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Render palette alone in Angular Color picker component

To render the **Palette** alone in ColorPicker, specify the [mode](#) property as **Palette**, and set the [modeSwitcher](#) property to **false**.

In the following sample, the [showButtons](#) property is disabled to hide the control buttons and it renders only the **Palette** area.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, ColorPickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <h4>Select Color</h4>
    <!-- To render Picker. -->
    <ejs-input ej2-colorpicker type="color" id="element"
mode="Palette" [modeSwitcher]="false" [showButtons]="false" /></div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

> To render **Picker** alone specify the [mode](#) property as 'Picker'.

Two way binding in Angular Color picker component

ColorPicker component supports two-way property binding.

The steps to perform two-way binding.

- Create [ColorPicker](#) component and binds the [value](#) property as like the below code snippet.

,

```
<input ej2-colorpicker type="color" class="form-control" id="colorpicker" required [(value)]="value"
name="colorpicker" />
```

,

- Create text box and bind the value using ngModel.

```
<input type="text" id="name" name="name" class="form-control" [(ngModel)]="value" />
```

- And name the same variable name in both color picker and text box. Which will help to view the two-way binding i.e. changing value in color picker will change the textbox value and vice versa.
- Initialize the value of the variable in component file, while will be bound to color picker and text box initially. The values will be changed synchronously while changing any one (color picker or text-box) value.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { FormsModule } from '@angular/forms'
import { Component, Input } from '@angular/core';
import { FormGroup } from '@angular/forms';
import { Browser } from '@syncfusion/ej2-base';
@Component({
  imports: [
    ColorPickerModule,
    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './template.html',
  styleUrls: ['./index.css']
})
export class AppComponent {
  private cValue: string = "#1de4d7";
  get value(): string {
    if (Browser.info.name === 'msie' && this.cValue.length > 7) {
      return this.cValue.substring(0, this.cValue.length - 2);
    } else {
      return this.cValue;
    }
  }
  @Input('value')
  set value(value: string) {
    this.cValue = value;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```


> By default, the selected color value returns 8 digit hex code in [value](#) property. Some browser like IE won't support the 8 digit hex code. In such case, you can use getter setter method to change the value to supported format as like the above sample.

Colorpicker in dropdownbutton in Angular Color picker component

This section explains about how to render the ColorPicker in DropDownButton. The [target](#) property of the DropDownButton helps to achieve this scenario. To know about the usage of [target](#) property refer to [Popup templating](#) section.

In the below sample, the color picker is rendered as inline type by setting [inline](#) property as `true` and the rendered color picker wrapper is passed as a [target](#) to the DropDownButton to achieve the above scenario.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerComponent, ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { DropDownButtonComponent, DropDownButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, ViewChild } from '@angular/core';
import { DropDownButtonComponent } from '@syncfusion/ej2-angular-splitbuttons';
import { ColorPickerEventArgs } from '@syncfusion/ej2-angular-inputs';
@Component({
  imports: [
    FormsModule, ColorPickerModule, DropDownButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <h4>Choose color</h4>
    <ejs-input ej2-colorpicker type="color" id="element" [inline]="true"
(change)="change($event)" />
    <ejs-button ej2-dropdownbutton #dropdownbtn id="dropdownbtn"
(open)="onOpen($event)" (beforeClose)="onClose($event)" target=".e-
colorpicker-wrapper" iconCss="e-dropdownbtn-preview"></ejs-button></div>`
})
export class AppComponent {
  @ViewChild('dropdownbtn')
  private ddb?: DropDownButtonComponent;
  public onOpen(args: any): void {
    args.element.parentElement.querySelector('.e-
cancel').addEventListener('click', this.closePopup.bind(this));
    this.open();
  }
  public onClose(args: any): void {
    args.element.parentElement.querySelector('.e-
cancel').removeEventListener('click', this.closePopup.bind(this));
  }
  public closePopup(): void {
    this.ddb?.toggle();
  }
}
```

```
// Triggers while selecting colors from color picker.
public change(args: ColorPickerEventArgs): void {
    if (this.ddb?.element && this.ddb.element.children.length > 0) {
        const firstChild = this.ddb.element.children[0] as HTMLElement;
        if (firstChild instanceof HTMLElement) {
            firstChild.style.backgroundColor = args.currentValue.hex;
        }
    }
    this.closePopup();
}

public open(): void {
    var zIndex = (document.getElementsByClassName('e-color-picker-
tooltip')[0] as HTMLElement).style.zIndex;
    var zIndexIntValue = parseInt(zIndex) + 2;
    var tooltip = (document.getElementsByClassName('e-color-picker-
tooltip')[0] as HTMLElement);
    if (tooltip) {
        tooltip.style.zIndex = zIndexIntValue.toString();
    }
}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customize colorpicker in Angular Color picker component

Custom palette

By default, the Palette will be rendered with default colors. To load custom colors in the palette, specify the colors in the [presetColors](#) property. To customize the color palette, add a custom class to palette tiles using [BeforeTileRender](#) event.

The following sample demonstrates the above functionalities.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
import { PaletteTileEventArgs, ColorPickerEventArgs } from '@syncfusion/ej2-
inputs';
import { addClass } from '@syncfusion/ej2-base';
@Component({
  imports: [
    FormsModule, ColorPickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: ` <div class="e-section-control">
```

```

        <div id="preview"></div>
        <h4>Select Color</h4>
        <ejs-input ej-colorpicker type="color" id="element"
value="#ba68c8" mode="Palette" [columns]="colCount" [inline]="true"
[modeSwitcher]="false" [showButtons]="false" [presetColors]="customColors"
(beforeTileRender)="tileRender($event)" (change)="onChange($event)" />
        </div>`
    })
    export class AppComponent {
        public tileRender(args: PaletteTileEventArgs): void {
            addClass([args.element], ['e-icons', 'e-custom-tile']);
        }
        // To specify number of columns to be rendered.
        public colCount: number = 4;
        // Triggers while selecting colors from palette.
        public onChange(args: ColorPickerEventArgs): void {
            (document.getElementById('preview') as
HTMLElement).style.backgroundColor = args.currentValue.hex;
        }
        // Triggers before rendering each palette tile.
        public customColors: { [key: string]: string[] } = {
            'custom1': ['#ef9a9a', '#e57373', '#ef5350',
                '#f44336', '#f48fb1', '#f06292',
                '#ec407a', '#e91e63', '#ce93d8',
                '#ba68c8', '#ab47bc', '#9c27b0',
                '#b39ddb', '#9575cd', '#7e57c2', '#673ab7'],
            'custom2': ['#9fa8da', '#7986cb', '#5c6bc0', '#3f51b5',
                '#90caf9', '#64b5f6', '#42a5f5', '#2196f3',
                '#81d4fa', '#4fc3f7', '#29b6f6', '#03a9f4',
                '#80ddea', '#4dd0e1', '#26c6da', '#00bcd4'],
            'custom3': ['#80cbc4', '#4db6ac', '#26a69a', '#009688',
                '#a5d6a7', '#81c784', '#66bb6a', '#4caf50',
                '#c5e1a5', '#aed581', '#9ccc65', '#8bc34a',
                '#e6ee9c',
                '#dce775', '#d4e157', '#cddc39']
        };
        ngOnInit(): void {
            (document.getElementById('preview') as
HTMLElement).style.backgroundColor = "#ba68c8"
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Hide input area from picker

By default, the input area will be rendered in ColorPicker. To hide the input area from it, add `e-hide-value` class to ColorPicker using the `cssClass` property.

In the following sample, the ColorPicker is rendered without input area.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, ColorPickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <h4>Choose Color</h4>
    <!-- To hide the value area. -->
    <ejs-input ej2-colorpicker type="color" id="element"
cssClass="e-hide-value" [modeSwitcher]="false" /></div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Custom handle

Color picker handle shape and UI can be customized. Here, we have customized the handle as **svg icon**. The same way you can customize the handle based on your requirement.

The following sample show the customized color picker handle.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerComponent, ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
import { OpenEventArgs } from '@syncfusion/ej2-angular-inputs';
@Component({
  imports: [
    FormsModule, ColorPickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <h4>Choose Color</h4>
    <!-- custom picker -->
    <ejs-input ej2-colorpicker type="color" id="element"
value="#344aee" cssClass="e-custom-picker" [modeSwitcher]="false"
(open)="onOpen($event)" /></div>`
```

```

    })
    export class AppComponent {
        onOpen(args: OpenEventArgs): void {
            (args.element.querySelector('.e-handler') as
            Element).classList.add('e-icons');
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Custom primary button

By default, the applied color will be updated in primary button of the color picker. You can customize that as **icon**.

In the following sample, the **picker** icon is added to primary button and using [change](#) event the selected color will be updated in bottom portion of the icon.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { Component, ViewChild } from '@angular/core';
import { addClass } from '@syncfusion/ej2-base';
import { ColorPickerEventArgs, ColorPickerComponent } from '@syncfusion/ej2-
angular-inputs';
@Component({
    imports: [
        FormsModule, ColorPickerModule
    ],
    standalone: true,
    selector: 'app-root',
    template: `<div class="e-section-control">
        <h4>Choose color</h4>
        <ejs-input ej2-colorpicker #colorpicker id="element"
        (change)="onChange($event)" /></div>`
})
export class AppComponent {
    @ViewChild('colorpicker')
    private colorPicker?: ColorPickerComponent;
    public onChange(args: ColorPickerEventArgs): void {
        const colorPickerElement = this.colorPicker?.element;
        if (colorPickerElement) {
            const nextSibling = colorPickerElement.nextElementSibling;
            if (nextSibling) {
                const selectedColorElement = nextSibling.querySelector('.e-
selected-color') as HTMLElement;
                if (selectedColorElement) {

```

```

        selectedColorElement.style.borderBottomColor =
args.currentValue.rgba;
    }
}
}
}
ngOnInit(): void {
    setTimeout(() => {
        addClass([(this.colorPicker!.element as
any).nextElementSibling.querySelector('.e-selected-color')], 'e-icons');
    }, 500);
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

> The Essential JS 2 provides a set of icons that can be loaded by applying **e-icons** class name to the element. You can also use third party icon to customize the primary button.

[Display hex code in input](#)

The color picker input element can be showcased in the place of primary button. The applied color hex code will be updated in the primary button input.

The following sample shows the color picker with input.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { Component, ViewChild } from '@angular/core';
import { ColorPickerComponent } from '@syncfusion/ej2-angular-inputs';
@Component({
  imports: [
    FormsModule, ColorPickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <h4>Choose color</h4>
    <ejs-input ej2-colorpicker #colorpicker id="element" readonly
/></div>`
})
export class AppComponent {
  @ViewChild('colorpicker')
  public colorPicker?: ColorPickerComponent;
  ngOnInit(): void {
    this.colorPicker!.element.type = 'text';
    this.colorPicker!.element.classList.add('e-input');
    setTimeout(() => {

```

```

        let proxy: any = this;
        let target: HTMLElement =
proxy.colorPicker.element.nextElementSibling;
        target.insertBefore(proxy.colorPicker.element,
target.children[1]);
        }, 500);
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Custom UI

The color picker UI can be customized in all possible ways. The following sample shows the excel like UI customization with help of SplitButton and Dialog component. In that by clicking the more colors option from color palette, the dialog contains color picker will open.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { Component, ViewChild } from '@angular/core';
import { ColorPickerEventArgs } from '@syncfusion/ej2-inputs';
import { EmitType, formatUnit } from '@syncfusion/ej2-base';
import { ItemModel, MenuEventArgs, BeforeOpenCloseMenuEventArgs } from '@syncfusion/ej2-splitbuttons';
import { ColorPickerComponent } from '@syncfusion/ej2-angular-inputs';
import { DialogComponent } from '@syncfusion/ej2-angular-popups';
import { SplitButtonComponent } from '@syncfusion/ej2-angular-splitbuttons';
@Component({
imports: [
FormsModule, ColorPickerModule, DialogModule, SplitButtonModule
],
standalone: true,
selector: 'app-root',
template: `<div class="e-section-control">
<h4>Select Color</h4>
<ejs-input ej2-colorpicker #colorpalette type="color"
id="colorpalette" cssClass="e-hide-palette" mode="Palette" [inline]="true"
[showButtons]="false" [modeSwitcher]="false"
(change)="paletteOnChange($event)" />
<ejs-splitbutton #splitbutton iconCss="e-icons e-font-icon"
[items]="items" (beforeClose)="onBeforeClose($event)"
(beforeItemRender)="beforeRender($event)" (select)="onSelect($event)"></ejs-
splitbutton>
<ejs-dialog id="modalDialog" #modalDialog cssClass="e-dlg-
picker" (open)="modalDlgOpen()" [isModal]="true" [width]="width"

```

```

[height]="height" [visible]="false" [target]='target'
[animationSettings]='animationSettings'
    (overlayClick)="dlgClose()">
    <ng-template #content>
        <ejs-input ejs-colorpicker #colorpicker type="color"
id="colorpicker" (change)="pickerOnChange($event)" [inline]="true"
[modeSwitcher]="false" />
    </ng-template>
</ejs-dialog>
</div>`
}))
export class AppComponent {
    @ViewChild('colorpalette')
    public colorPalette?: ColorPickerComponent;
    @ViewChild('colorpicker')
    public colorPicker?: ColorPickerComponent;
    @ViewChild('modalDialog')
    public modalDialog?: DialogComponent;
    @ViewChild('splitbutton')
    public splitBtn?: SplitButtonComponent;
    public items: ItemModel[] = [
        {
            text: ''
        },
        {
            text: "More Colors...",
            iconCss: "e-switcher"
        }
    ];
    public target: string = ".wrap";
    public width: string = '270px';
    public height: string = '336px';
    public animationSettings: Object = { effect: 'Zoom' };
    public beforeRender(args: MenuEventArgs): void {
        if (args.item.text === "") {
            this.colorPalette!.cssClass = "";
            this.colorPalette?.dataBind();
            this.colorPalette?.refresh();
            args.element.appendChild((this.colorPalette as
any).element.parentElement);
        }
    }
    public modalDlgOpen: EmitType<object> = () => {
        this.colorPicker?.refresh();
        ((this.colorPicker as any)?.element.parentElement).querySelector('.e-ctrl-btn .e-cancel').addEventListener('click', this.dlgClose);
    }
    public onBeforeClose(args: BeforeOpenCloseMenuEventArgs): void {
        document.body.appendChild((this.colorPalette as
any).element.parentElement);
        this.colorPalette!.cssClass = "e-hide-palette";
        this.colorPalette?.dataBind();
    }
    public dlgClose: any = (args: any) => {
        this.modalDialog?.hide();
    }
    public pickerOnChange(args: ColorPickerEventArgs): void {

```



```

        this.paletteOnChange(args);
        this.modalDialog?.hide();
    }
    public paletteOnChange(args: ColorPickerEventArgs): void {
        ((this.splitBtn as SplitButtonComponent).element.querySelector(".e-
font-icon") as HTMLElement).style.borderBottomColor = args.currentValue.rgba;
    }
    public onSelect(args: MenuEventArgs): void {
        if (args.item.text === 'More Colors...') {
            this.modalDialog?.show();
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Handle no color support in Angular Color picker component

The ColorPicker component supports no color functionality. By clicking the no color tile from palette, the selected color becomes **empty** and considered as no color has been selected from color picker.

Default no color

To achieve this, set [noColor](#) property as **true**.

In the following sample, the first tile of the color palette represents the no color tile. By clicking the no color tile you can achieve the above functionalities.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { Component, OnInit } from '@angular/core';
import { ColorPickerEventArgs } from '@syncfusion/ej2-angular-inputs';
@Component({
  imports: [
    FormsModule, ColorPickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id="preview"></div>
    <h4>Choose Color</h4>
    <ejs-input ej2-colorpicker type="color" id="element"
mode="Palette" value="#ba68c8" [showButtons]="false"
(change)="onChange($event)" [modeSwitcher]="false" [noColor]="true" />
    </div>`
})
export class AppComponent implements OnInit {
  public preview?: HTMLElement;

```

```
// Triggers while color value changes.
public onChange(args: ColorPickerEventArgs): void {
    this.preview!.style.backgroundColor = args.currentValue.hex;
    if (args.currentValue.hex) {
        this.preview!.textContent = args.currentValue.hex;
    } else {
        this.preview!.textContent = "No color"
    }
}
ngOnInit(): void {
    this.preview = document.getElementById('preview') as HTMLElement;
    this.preview.style.backgroundColor = "#ba68c8";
    this.preview.textContent = "#ba68c8";
}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

If the [noColor](#) property is enabled, make sure to disable the [modeswitcher](#) property.

Custom no color

The following sample show the color palette with custom no color option.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { Component, ViewChild, OnInit } from '@angular/core';
import { ColorPickerComponent } from '@syncfusion/ej2-angular-inputs';
import { SplitButtonComponent } from '@syncfusion/ej2-angular-splitbuttons';
@Component({
  imports: [
    FormsModule, ColorPickerModule, SplitButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <ul id="target" tabindex="0">
      <li class="e-item e-palette-item">
        <ejs-input ej2-colorpicker #palette type="color"
value="#f44336" id="element" mode="Palette" [inline]="true"
(change)="onChange($event)" (beforeTileRender)="beforeRender($event)"
[modeSwitcher]="false" [showButtons]="false" [columns]="column"
[presetColors]="customColors" />
      </li>
      <li class="e-item" tabindex="-1"
(click)="noColorClicked()">
        <span class="e-menu-icon e-nocolor"></span>
```

```

        No color
    </li>
</ul>
<div>
    <div id="preview"></div>
    <h4>Select Color</h4>
    <ejs-splitbutton #splitbtn iconCss="e-cp-icons e-picker-
icon" target="#target"></ejs-splitbutton>
</div>
</div>`
}))
export class AppComponent implements OnInit {
    @ViewChild("palette")
    public palette?: ColorPickerComponent;
    @ViewChild("splitbtn")
    public splitBtn?: SplitButtonComponent;
    public customColors: { [key: string]: string[] } = {
        'custom': ['#f44336', '#e91e63', '#9c27b0', '#673ab7', '#2196f3',
        '#03a9f4', '#00bcd4',
        '#009688', '#8bc34a', '#cddc39', '#ffeb3b', '#ffc107']
    };
    public column: number = 4;
    public preview?: HTMLElement;
    public beforeRender(args: any): void {
        args.element.classList.add('e-custom-tile');
    }
    public noColorClicked(): void {
        //sets color picker value property to null
        this.palette?.setProperties({ 'value': "" }, true);
        (document.querySelector('.e-split-btn .e-picker-icon') as
        HTMLElement).style.borderBottomColor = "transparent";
        this.preview!.textContent = "No color"
        this.preview!.style.backgroundColor = "transparent";
    }
    // Triggers while color value changes.
    public onChange(args: any): void {
        (document.querySelector(".e-split-btn .e-picker-icon") as
        HTMLElement).style.borderBottomColor = args.currentValue.hex;
        this.preview!.style.backgroundColor = args.currentValue.hex;
        this.preview!.textContent = args.currentValue.hex;
        if (this.splitBtn?.element.getAttribute("aria-expanded")) {
            this.splitBtn.toggle();
            this.splitBtn.element.focus();
        }
    }
    ngOnInit(): void {
        this.preview = document.getElementById('preview') as HTMLElement;
        this.preview.style.backgroundColor = "#f44336";
        this.preview.textContent = "#f44336";
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';

```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Disabled in Angular Color picker component

To achieve disabled state in ColorPicker, set the [disabled](#) property to `true`. The ColorPicker pop-up cannot be accessed in disabled state.

The following example shows the `disabled` state of ColorPicker component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerModule } from '@syncfusion/ej2-angular-inputs'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, ColorPickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <h4>Disabled State</h4>
    <!-- To disable ColorPicker. -->
    <ejs-input ej-colorpicker type="color" id="element"
[disabled]="true" /></div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Ej1 api migration in Angular Color picker component

This article describes the API migration process of ColorPicker component from Essential JS 1 to Essential JS 2.

Properties

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Default | **property:** *value*

 <input ej-colorpicker id="colorpicker" value="#278787" /> | **property:** *value*

 <input ej-colorpicker id="colorpicker" value="#278787" /> |

| Inline mode color picker | **property:** *displayInline*

 <input ej-colorpicker id="colorpicker" [displayInline]="true" /> | **property:** *inline*

 <input ej-colorpicker id="colorpicker" [inline]="true" /> |

| Adding custom class | **property:** *cssClass*

 <input ej-colorpicker id="colorpicker" cssClass="custom-class" /> | **property:** *cssClass*

 <input ej-colorpicker id="colorpicker" cssClass="custom-class" /> |

| Disable the ColorPicker component | **property:** *enabled*

 <input ej-colorpicker id="colorpicker" [enabled]="false" /> | **property:** *disabled*

 <input ej-colorpicker id="colorpicker" [disabled]="true" /> |

| To display custom text in button elements | **property:** *buttonText*

 <input ej-colorpicker id="colorpicker" [buttonText]="buttonText" />
 buttonText: Object = { apply: "Apply", cancel: "Cancel", swatches: "Swatches" }; | Not Applicable |

| To display customized text or content when mouse over the color picker elements | **property:** *tooltipText*

 <input ej-colorpicker id="colorpicker" [tooltipText]="tooltipText" />
 tooltipText: Object = { switcher: "Switch", currentColor: "New Color", selectedColor: "Old Color" }; | Not Applicable |

| Disable / hide opacity | **property:** *enableOpacity*

 <input ej-colorpicker id="colorpicker" [enableOpacity]="false" /> | **property:** *enableOpacity*

 <input ej-colorpicker id="colorpicker" [enableOpacity]="false" /> |

| ColorPicker Button mode | **property:** *buttonMode*

 <input ej-colorpicker id="colorpicker" buttonMode="Dropdown" /> | Not Applicable |

| To show / hide the control (apply / cancel) buttons | **property:** *showApplyCancel*

 <input ej-colorpicker id="colorpicker" [showApplyCancel]="false" /> | **property:** *showButtons*

 <input ej-colorpicker id="colorpicker" [showButtons]="false" /> |

| To show / hide the clear button | **property:** *showClearButton*

 <input ej-colorpicker id="colorpicker" [showClearButton]="false" /> | Not Applicable |

| Show / hide the mode (picker / palette) switcher | **property:** *showSwitcher*

 <input ej-colorpicker id="colorpicker" [showSwitcher]="false" /> | **property:** *modeSwitcher*

 <input ej-colorpicker id="colorpicker" [modeSwitcher]="false" /> |

| To show / hide the preview area | **property:** *showPreview*

 <input ej-colorpicker id="colorpicker" [showPreview]="false" /> | Not Applicable |

| To show / hide the recent selected color list | **property:** *showRecentColors*

 <input ej-colorpicker id="colorpicker" [showRecentColors]="true" /> | Not Applicable |

| To show / hide the color picker slider tooltip | **property:** *showTooltip*

 <input ej-colorpicker id="colorpicker" [showTooltip]="false" /> | Not Applicable |

| Custom icon in dropdown control color area | **property:** *toolIcon*

 <input ej-colorpicker id="colorpicker" toolIcon="e-font-icon" /> | Not Applicable |

| ColorPicker mode | **property:** *modelType*

 <input ej-colorpicker id="colorpicker" modelType="Palette" /> | **property:** *mode*

 <input ej-colorpicker id="colorpicker" mode="Palette" /> |

| Opacity value | **property:** *opacityValue*

 <input ej-colorpicker id="colorpicker" opacityValue=80 /> | Not Applicable |

| Number of columns in color palette | **property:** *columns*

 <input ej-colorpicker id="colorpicker" columns=10 /> | **property:** *columns*

 <input ej-colorpicker id="colorpicker" columns=15 /> |

| Custom colors | **property:** *palette*

 <input ej-colorpicker id="colorpicker" palette="CustomPalette" modelType="Palette" [custom]="colors" />
 colors: Array<any> = ["ffffff", "ffccff", "ff99ff", "ff66ff", "ff33ff", "ff00ff", "ccffff", "ccccff", "cc99ff", "cc66ff", "cc33ff", "cc00ff", "99ffff", "99ccff", "9999ff", "9966ff", "9933ff", "9900ff", "ffffcc", "ffcccc"]; | **property:** *presetColors*

 <input ej-colorpicker id="colorpicker" mode="Palette" [presetColors]="colors" />
 colors: { [key: string]: string[] } = {
 'custom': ["ffffff", "ffccff", "ff99ff", "ff66ff", "ff33ff", "ff00ff", "ccffff", "ccccff", "cc99ff", "cc66ff", "cc33ff", "cc00ff", "99ffff", "99ccff", "9999ff", "9966ff", "9933ff", "9900ff", "ffffcc", "ffcccc"]
 }; |

| Rendering palette from the predefined set of palettes | **property:** *presetType*

 <input ej-colorpicker id="colorpicker" modelType="Palette" presetType="FlatColors" /> | Not Applicable |

| No color option in color palette | Not Applicable | **property:** *noColor*

 <input ej-colorpicker id="colorpicker" [noColor]="true" [modeSwitcher]="false" mode="Palette" /> |

| Localization | **property:** *locale*

 ej.ColorPicker.Locale["zh-CN"] = {
 buttonText: {
 apply: "应用",
 cancel: "取消",
 swatches: "色板"
 },
 tooltipText: {
 switcher: "切换器",
 addButton: "添加颜色",
 basic: "基本"
 }
 }
 <input ej-colorpicker id="colorpicker" locale="zh-CN" /> | **property:** *locale*

 L10n.load({
 'ar': {
 "colorpicker": {
 "Apply": "تطبيق",
 "Cancel": "إلغاء",
 "ModeSwitcher": "مفتاح كهربائي الوضع"
 }
 }
 });
 <input ej-colorpicker id="element" locale="ar" /> |

| Right to left | **property:** *enableRTL*

 <input ej-colorpicker id="colorpicker" [enableRTL]="true" /> | **property:** *enableRtl*

 <input ej-colorpicker id="colorpicker" [enableRtl]="true" /> |

Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Method to open color picker popup | **Method:** *show*

 <input ej-colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.show(); | **Method:** *toggle*

 <input ej-colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.toggle(); |

| Method to close color picker popup | **Method:** *hide*

 <input ej-colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.hide(); | **Method:** *toggle*

 <input ej-

colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.toggle(); |

| Enable the color picker control | **Method:** *enable*

 <input ej-colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.enable(); | Not Applicable |

| Disables the color picker control | **Method:** *disable*

 <input ej-colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.disable(); | Not Applicable |

| Method returns the selected color value as hex code | **Method:** *getValue*

 <input ej-colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.getValue(); | **Method:** *getValue*

 <input ej-colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.getValue(); |

| Method returns the selected color value in RGB format | **Method:** *getColor*

 <input ej-colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.getColor(); | **Method:** *getValue*

 <input ej-colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.getValue(null, 'RGB'); |

| Method convert the color value from hexCode to RGB | **Method:** *hexCodeToRGB*

 <input ej-colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.hexCodeToRGB("#278787"); | **Method:** *getValue*

 <input ej-colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.getValue("#278787", 'RGB'); |

| Method convert the color value from RGB to Hex code | **Method:** *RGBToHEX*

 <input ej-colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.RGBToHEX({r:38,g:133,b:133}); | **Method:** *getValue*

 <input ej-colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.getValue("rgb(38,133,133)", 'Hex'); |

| Method convert the color value from RGB to HSV | **Method:** *RGBToHSV*

 <input ej-colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.RGBToHSV({h:230,s:98,v:98}); | **Method:** *getValue*

 <input ej-colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.getValue("rgb(180,71.1,52.9)", 'HSV'); |

| Method convert the color value from HSV to RGB | **Method:** *HSVToRGB*

 <input ej-colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.HSVToRGB({h:230,s:98,v:98}); | **Method:** *getValue*

 <input ej-colorpicker #colorpicker id="colorpicker" />
 @ViewChild('colorpicker')
 public colorPickerObj: ColorPickerComponent;
 this.colorPickerObj.getValue("hsv(180,71.1,52.9)", 'RGB'); |

Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Event triggers before opening the ColorPicker popup | Not Applicable | **Event:** *beforeOpen*

<input ej-colorpicker id="colorpicker" (beforeOpen)="beforeOpen(\$event)" />

beforeOpen(args) {
 / code block */
 } |

| Event triggers before closing the ColorPicker popup | Not Applicable | **Event:** *beforeClose*

<input ej-colorpicker id="colorpicker" (beforeClose)="beforeClose(\$event)" />

beforeClose(args) {
 / code block */
 } |

| Event triggers after opening the ColorPicker popup | **Event:** *open*

 <input ej-colorpicker

id="colorpicker" (open)="open(\$event)" />
 open(args) {
 / code block /

 } | **Event:** *open*

 <input ej-colorpicker id="colorpicker" (open)="open(\$event)" />

 open(args) {
 / code block /
 } |

| Event triggers after closing the ColorPicker popup | **Event:** *close*

 <input ej-colorpicker

id="colorpicker" (close)="close(\$event)" />
 close(args) {
 / code block */

 } | Not Applicable |

| Event triggers once the component rendering is completed | **Event:** *create*

 <input ej-

colorpicker id="colorpicker" (create)="create(\$event)" />
 create(args) {

/ code block /
 } | **Event:** *created*

 <input ej-colorpicker id="colorpicker"

(created)="created()" />
 created() {
 / code block /
 } |

| Event triggers once the color picker control is destroyed | **Event:** *destroy*

 <input ej-

colorpicker id="colorpicker" (destroy)="destroy(\$event)" />
 destroy(args) {

 / code block */
 } | Not Applicable |

| Event triggers before Switching between Picker / Palette mode | Not Applicable | **Event:**

beforeModeSwitch

 <input ej-colorpicker id="colorpicker"

(beforeModeSwitch)="beforeModeSwitch(\$event)" />
 beforeModeSwitch(args) {

 / code block */
 } |

| Event triggers after color value has been selected | **Event:** *select*

 <input ej-colorpicker

id="colorpicker" (select)="select(\$event)" />
 select(args) {
 / code block /

 } | **Event:** *select*

 <input ej-colorpicker id="colorpicker" (select)="select(\$event)" />

 select(args) {
 / code block /
 } |

| Event triggers after color value has been changed | **Event:** *change*

 <input ej-colorpicker

id="colorpicker" (change)="change(\$event)" />
 change(args) {
 / code

block /
 } | **Event:** *change*

 <input ej-colorpicker id="colorpicker"

(change)="change(\$event)" />
 change(args) {
 / code block /
 } |

ComboBox

Getting started with Angular Combo box component

This section explains how to create a simple [Link to the Video](#) component and configure its available functionalities in Angular.

To get started quickly with angular ComboBox component using angular CLI, you can check the video below.

Dependencies

The following list of dependencies are required to use the Angular ComboBox component in your application.

```
`javascript
|-- @syncfusion/ej2-angular-dropdowns
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-angular-base
|-- @syncfusion/ej2-dropdowns
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-buttons
`,`
```

Setup angular environment

Angular provides the easiest way to set angular CLI projects using [Angular CLI](#) tool.

Install the CLI application globally to your machine.

```
`bash
npm install -g @angular/cli
`,`
```

Create a new application

```
`bash
ng new syncfusion-angular-combobox
`,`
```

By default, it install the CSS style base application. To setup with SCSS, pass --style=scss argument on create project.

Example code snippet.

```
`bash
ng new syncfusion-angular-combobox --style=scss
`,`
```

Navigate to the created project folder.

```
`bash
```

```
cd syncfusion-angular-combobox
```

Installing Syncfusion ComboBox package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(`>=20.2.36`) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-dropdowns](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-dropdowns --save
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-dropdowns@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-dropdowns@ngcc --save
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-dropdowns:"20.2.38-ngcc"
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Registering ComboBox module

Import ComboBox module into Angular application(app.module.ts) from the package `@syncfusion/ej2-angular-dropdowns`.

```
`javascript
import { NgModule } from '@angular/core';
```

```
import { BrowserModule } from '@angular/platform-browser';
// import the ComboBoxModule for the ComboBox component
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns';
import { AppComponent } from './app.component';
@NgModule({
  //declaration of ej2-angular-dropdowns module into NgModule
  imports: [ BrowserModule, ComboBoxModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Adding CSS reference

The following CSS files are available in `../node_modules/@syncfusion` package folder.

This can be referenced in `[src/styles.css]` using following code.

```
`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-angular-dropdowns/styles/material.css';
```

Adding ComboBox component

Modify the template in `[src/app/app.component.ts]` file to render the Angular ComboBox component. Add the Angular ComboBox by using `<ejs-combobox>` selector in `template` section of the `app.component.ts` file.

```
`javascript
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  // specifies the template string for the ComboBox component
  template: <ejs-combobox id='comboelement'></ejs-combobox>
```

```

})
export class AppComponent { }
`

```

Binding data source

After initializing, populate the ComboBox with data using the [dataSource](#) property. Here, an array of string values passed to ComboBox component.

```

`typescript
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  // specifies the template string for the ComboBox component
  template: <ejs-combobox id='comboelement' [dataSource]='data'></ejs-combobox>
})
export class AppComponent {
  constructor() {
  }
  // defined the array of data
  public data: string[] = ['Cricket', 'Football', 'Rugby', 'Snooker', 'Tennis'];
}
`

```

Running the application

After completing the configuration required to render a basic ComboBox, run the following command to display the output in your default browser.

```

`
ng serve
`

```

The following example illustrates the output in your browser.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],

```

```
standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component with
  dataSource
  template: `<ejs-combobox id='comboelement' [dataSource]='data'
placeholder = 'Select a game'></ejs-combobox>`
})
export class AppComponent {
  constructor() {
  }
  // defined the array of data
  public data: string[] = ['Cricket', 'Football', 'Rugby', 'Snooker',
'Tennis'];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Custom values

The ComboBox allows the user to give input as custom value which is not required to present in predefined set of values. By default, this support is enabled by [allowCustom](#) property. In this case, both text field and value field considered as same. The custom value will be sent to post back handler when a form is about to be submitted.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component with
  dataSource
  template: `<ejs-combobox id='comboelement' [dataSource]='sportsData'
[fields]=fields allowCustom=true placeholder = 'Select a game'></ejs-
combobox>`
})
export class AppComponent {
  constructor() {
  }
  public fields: Object = {text: 'Game', value: 'Id'};
  // defined the array of data
  public sportsData: { [key: string]: Object }[] = [
    { Id: 'game1', Game: 'Badminton' },
```

```

        { Id: 'game2', Game: 'Football' },
        { Id: 'game3', Game: 'Tennis' }
    ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Configure the popup list

By default, the width of the popup list automatically adjusts according to the ComboBox input element's width, and the height of the popup list has '300px'.

The height and width of the popup list can also be customized using the [popupHeight](#) and [popupWidth](#) property respectively.

In the following sample, popup list's width and height are configured.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component with value property
  template: `<ejs-combobox id='comboelement' #samples [dataSource]='data'
placeholder='Select a game' popupHeight='200px' popupWidth='250px'></ejs-combobox>`,
})
export class AppComponent {
  constructor() {
  }
  // define the array of data
  public data: string[] = ['Cricket', 'Football', 'Golf', 'Rugby', 'Snooker', 'Tennis'];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Two-way binding

In ComboBox, the `value` property supports two-way binding functionality. The following example demonstrates how to work the two-way binding functionality in ComboBox.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component and
  // input element for checking the two-way binding support using value
  property
  template: `
    <ejs-combobox id='comboelement' [dataSource]='data' [(value)]='value'
    placeholder="Select a game"></ejs-combobox>
    <div style='margin-top: 50px'>
      <input type="text" [(ngModel)]="value"
    style='width:245px;height:25px' />
    </div>
  `
})
export class AppComponent {
  constructor() {
    // defined the array of complex data
    public data: string[] = [ 'Badminton', 'Football', 'Rugby', 'Snooker',
    'Tennis' ];
    // set a value to pre-select
    public value: string = 'Badminton';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

You can refer to our [Angular ComboBox](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular ComboBox example](#) that shows how to render the ComboBox in Angular.

See Also

- [How to bind the data](#)

Data binding in Angular Combo box component

The ComboBox loads the data either from local data sources or remote data services using the [dataSource](#) property. It supports the data type of `array` or `DataManager`.

The ComboBox also supports different kinds of data services such as OData, OData V4, and Web API, and data formats such as XML, JSON, and JSONP with the help of `DataManager` adaptors.

Fields	Type	Description
text	string	Specifies the display text of each list item.
value	number or string	Specifies the hidden data value mapped to each list item that should contain a unique value.
groupBy	string	Specifies the category under which the list item has to be grouped.
iconCss	string	Specifies the icon class of each list item.

When binding complex data to the ComboBox, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Binding local data

Local data can be represented in two ways as described below.

1. Array of simple data

The ComboBox has support to load array of primitive data such as strings and numbers. Here, both value and text field act the same.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component
  template: `<ejs-combobox id='comboelement' #samples [dataSource]='data'
[placeholder]='text'></ejs-combobox>`
})
export class AppComponent {
  constructor() {
  }
  // defined the array of data
```



```
public data: string[] = ['Badminton', 'Basketball', 'Cricket', 'Golf',
'Hockey', 'Rugby'];
// set placeholder text to ComboBox input element
public text: string = 'Select a game';
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

2. Array of JSON data

The ComboBox can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, **Id** column and **Game** column from complex data have been mapped to the **value** field and **text** field, respectively.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component with change event
  template: `<ejs-combobox id='comboelement' #samples [dataSource]='data' [fields]='fields' [placeholder]='text'></ejs-combobox>`
})
export class AppComponent {
  constructor() {
  }
  // defined the array of data
  public data: { [key: string]: Object }[] = [ { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' }, { Id: 'game3', Game: 'Tennis' } ];
  // maps the appropriate column to fields property
  public fields: Object = { text: 'Game', value: 'Id' };
  //set the placeholder to ComboBox input
  public text: string = "Select a game";
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

3. Array of Complex data

The ComboBox can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, Code.Id column and 'Country.Name' column from complex data have been mapped to the value field and text field, respectively.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component with change event
  template: `<ejs-combobox id='comboelement' #samples [dataSource]='data' [fields]='fields' [placeholder]='text'></ejs-combobox>`
})
export class AppComponent {
  constructor() {
  }
  // defined the array of data
  public data: { [key: string]: Object }[] = [
    { Country: { Name: 'Australia' }, Code: { Id: 'AU' } },
    { Country: { Name: 'Bermuda' }, Code: { Id: 'BM' } },
    { Country: { Name: 'Canada' }, Code: { Id: 'CA' } },
    { Country: { Name: 'Cameroon' }, Code: { Id: 'CM' } },
    { Country: { Name: 'Denmark' }, Code: { Id: 'DK' } },
    { Country: { Name: 'France' }, Code: { Id: 'FR' } }
  ];
  // maps the appropriate column to fields property
  public fields: Object = { text: 'Country.Name', value: 'Code.Id' };
  //set the placeholder to ComboBox input
  public text: string = "Select a country";
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Binding remote data

The ComboBox supports retrieval of data from remote data services with the help

of **DataManager** component. The **Query** property allows is used to fetch data from the database and bind it to the ComboBox.

In the following sample, displayed first 6 contacts from **customer** table of **Northwind** Data Service.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data'
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component with change event
  template: `<ejs-combobox id='comboelement' #samples [dataSource]='data' [fields]='fields' [placeholder]='text' [query]='query' [sortOrder]='sorting'></ejs-combobox>`
})
export class AppComponent {
  constructor() {
  }
  //bind the DataManager instance to dataSource property
  public data: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  });
  // maps the appropriate column to fields property
  public fields: Object = { text: 'ContactName', value: 'CustomerID' };
  //bind the Query instance to query property
  public query: Query = new
  Query().from('Customers').select(['ContactName', 'CustomerID']).take(6);
  //set the placeholder to ComboBox input
  public text: string = "Select a customer";
  //sort the result items
  public sorting: string = 'Ascending';
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Data binding using Async pipe

An **Observable** is used extensively by Angular since it provide significant benefits over techniques for event handling, asynchronous programming, and handling multiple values.

ComboBox data can be consumed from an **Observable** object by piping it through an **async** pipe. The **async** pipe is used to subscribe the observable object and resolve with the latest value emitted by it.

[app.component.ts]

`ts

```
import { Component } from '@angular/core';
```

```
import { Observable } from 'rxjs';
```

```
import { map } from 'rxjs/operators';
```

```
import { HttpClient } from '@angular/common/http';
```

```
@Component({
```

```
  selector: 'app-root',
```

```
  // specifies the template string for the ComboBox component with dataSource
```

```
  template: `<ejs-combobox id='customers2' formControlName="skillname" name="skillname"
#remote2 [dataSource]='data | async' [fields]='remoteFields'
[placeholder]='remoteWaterMark' ></ejs-combobox >`,
```

```
})
```

```
export class AppComponent {
```

```
  constructor(private http: HttpClient){
```

```
    this.data=this.http.get<[[key:
```

```
string]:object;]]>('https://services.odata.org/V4/Northwind/Northwind.svc/Customers').pipe(
```

```
map((results: { [key: string]: any }) => {
```

```
  return results['value'];
```

```
})
```

```
};
```

```
}
```

```
public data: Observable<any>;
```

```
// maps the remote data column to fields property
```

```
public remoteFields: Object = { value: 'CustomerID' };
```

```
// set the placeholder to ComboBox input element
```

```
public remoteWaterMark: string = 'Select a customer';
```

```

}
,

[app.module.ts]
`ts
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
import { DropDownListModule, ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns';
import { AppComponent } from './app.component';
import { DialogModule } from '@syncfusion/ej2-angular-popups';
import { ReactiveFormsModule } from '@angular/forms';
@NgModule({
  imports: [
    BrowserModule,
    FormsModule,
    DropDownListModule,
    ComboBoxModule,
    DialogModule,
    HttpClientModule,
    ReactiveFormsModule
  ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
,

[main.ts]
`ts
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
enableProdMode();

```

```
platformBrowserDynamic().bootstrapModule(AppModule);
```

,

[View Sample in Github](#)

See Also

- [How to achieve cascading](#)
- [How to load data using template](#)
- [How to group the data using header](#)
- [How to filter the bound data](#)

Value binding in ComboBox Component

Value binding in the ComboBox control allows you to associate data values with each list item. This facilitates managing and retrieving selected values efficiently. The ComboBox component provides flexibility in binding both primitive data types and complex objects.

Primitive Data Types

The ComboBox control provides flexible binding capabilities for primitive data types like strings and numbers. You can effortlessly bind local primitive data arrays, fetch and bind data from remote sources, and even custom data binding to suit specific requirements. Bind the value of primitive data to the [value](#) property of the ComboBox.

Primitive data types include:

- String
- Number
- Boolean
- Null

The following sample shows the example for preselect values for primitive data type

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component } from '@angular/core';
import { ComboBoxComponent } from '@syncfusion/ej2-angular-dropdowns';
@Component({
  imports: [
    FormsModule, ComboBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the virtual-scroll url path
  templateUrl: 'virtual-scroll.html'
})
export class AppComponent {
  // defined the array of data
  public records: string[] = [];
  constructor() {
```

```

        this.records = ["Item 1", "Item 2", "Item 3", "Item 4", "Item 5",
        "Item 6", "Item 7", "Item 8", "Item 9", "Item 10"];
    }
    // maps the appropriate column to fields property
    public fields: object = { text: 'text', value: 'id' };
    public value = "Item 11";
    // set the placeholder to AutoComplete input
    public waterMark: string = 'e.g. Item 1';
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Object Data Types

In the ComboBox control, object binding allows you to bind to a dataset of objects. When [Link to the Video](#) is enabled, the value of the control will be an object of the same type as the selected item in the [value](#) property. This feature seamlessly binds arrays of objects, whether sourced locally, retrieved from remote endpoints, or customized to suit specific application needs.

The following sample shows the example for preselect values for object data type

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component } from '@angular/core';
import { ComboBoxComponent } from '@syncfusion/ej2-angular-dropdowns';
@Component({
  imports: [
    FormsModule, ComboBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the virtual-scroll url path
  templateUrl: 'virtual-scroll.html'
})
export class AppComponent {
  // defined the array of data
  public records: { [key: string]: Object }[] = [];
  constructor() {
    for (let i: number = 1; i <= 150; i++) {
      const item: { [key: string]: Object } = {
        id: 'id' + i,
        text: `Item ${i}`,
      };
      this.records.push(item);
    }
  }
  // maps the appropriate column to fields property

```

```
public fields: object = { text: 'text', value: 'id' };
public value = {id: 'id11', text: 'Item 11'};
// set the placeholder to AutoComplete input
public waterMark: string = 'e.g. Item 1';
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Templates in Angular Combo box component

The ComboBox has been provided with several options to customize each list items, group title, header, and footer elements.

To get started quickly with templates in angular ComboBox component, you can check the video below.

Item template

The content of each list item within the ComboBox can be customized with the help of [itemTemplate](#) property.

In the following sample, each list item is split into two columns to display relevant data's.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component } from '@angular/core';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data'
@Component({
  imports: [
    FormsModule, ComboBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template url path
  templateUrl: 'template.html'
})
export class AppComponent {
  height: any;
  constructor() {
  }
  //bind the DataManager instance to dataSource property
  public data: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  });
  // maps the appropriate column to fields property
  public fields: Object = { text: 'FirstName', value: 'EmployeeID' };
  //bind the Query instance to query property
```



```
public query: Query = new Query().from('Employees').select(['FirstName',
'City','EmployeeID']).take(6);
//set the placeholder to ComboBox input
public text: string = "Select an employee";
//sort the result items
public sorting: string = 'Ascending';
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

TEMPLATE.HTML

```
<div id="wrapper">
  <div id='content' style="margin: 40px auto 0; width:250px;">
    <!-- specifies the template string for the ComboBox component-->
    <ejs-combobox id='combobox-template' [dataSource]='data'
[fields]='fields' [sortOrder]='sorting' [query]='query'
[popupHeight]='height' [placeholder]='text' [itemTemplate]='itemTemplate'>
      <ng-template #itemTemplate="" let-data="">
        <!--set the value to itemTemplate property-->
        <span><span class='name'> {{data.FirstName}}</span><span
class ='city'>{{data.City}}</span></span>
      </ng-template>
    </ejs-combobox>
  </div>
</div>
```

Group template

The group header title under which appropriate sub-items are categorized can also be customize with the help of [groupTemplate](#) property.

This template is common for both inline and floating group header template.

In the following sample, employees are grouped according to their city.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component } from '@angular/core';
import { Query, Predicate, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data'
@Component({
  imports: [
    FormsModule, ComboBoxModule
  ],
  standalone: true,
  selector: 'app-root',
```

```
// specifies the template url path
templateUrl: 'template.html'
})
export class AppComponent {
  constructor() {
  }
  //bind the DataManager instance to dataSource property
  public data: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  });
  // form predicate to fetch the grouped data
  public groupPredicate = new Predicate('City',
    'equal', 'london').or('City', 'equal', 'seattle');
  // maps the appropriate column to fields property
  public fields: Object = { text: 'FirstName', value: 'EmployeeID',
groupBy: 'City' };
  //bind the Query instance to query property
  public query: Query = new Query().from('Employees').select(['FirstName',
    'City', 'EmployeeID']).take(5).where(this.groupPredicate);
  //set the placeholder to ComboBox input
  public text: string = "Select an employee";
  //sort the result items
  public sorting: string = 'Ascending';
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

TEMPLATE.HTML

```
<div id="wrapper">
  <div id='content' style="margin: 40px auto 0; width:250px;">
    <!-- specifies the template string for the ComboBox component-->
    <ejs-combobox id='comboelement' #samples [dataSource]='data'
[fields]='fields' [sortOrder]='sorting' [placeholder]='text'
[groupTemplate]='groupTemplate' [query]='query'>
      <ng-template #groupTemplate="" let-data="">
        <!--set the value to groupTemplate property-->
        <strong>{{data.City}}</strong>
      </ng-template>
    </ejs-combobox>
  </div>
</div>
```

Header template

The header element is shown statically at the top of the popup list items within the ComboBox, and any custom element can be placed as a header element using the

[headerTemplate](#) property.

In the following sample, the list items and its headers are designed and displayed as two columns similar to multiple columns of the grid.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component } from '@angular/core';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
@Component({
  imports: [
    FormsModule,ComboBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template url path
  templateUrl: 'template.html'
})
export class AppComponent {
  constructor() {
  }
  //bind the DataManager instance to dataSource property
  public data: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  });
  // maps the appropriate column to fields property
  public fields: Object = { text: 'FirstName', value: 'EmployeeID' };
  //bind the Query instance to query property
  public query: Query = new Query().from('Employees').select(['FirstName',
'City','EmployeeID']).take(6);
  //set the placeholder to ComboBox input
  public text: string = "Select an employee";
  //sort the result items
  public sorting: string = 'Ascending';
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

TEMPLATE.HTML

```
<div id="wrapper">
  <div id='content' style="margin: 40px auto 0; width:250px;">
    <!-- specifies the template string for the ComboBox component-->
```

```
<ejs-combobox id='comboelement' #samples [dataSource]='data'
[fields]='fields' [sortOrder]='sorting' [placeholder]='text' [query]='query'
[headerTemplate]='headerTemplate' [itemTemplate]='itemTemplate'>
  <ng-template #itemTemplate="" let-data="">
    <!--set the value to itemTemplate property-->
    <span class='item'><span class='name'>
      {{data.FirstName}}</span><span class='city'>{{data.City}}</span></span>
    </ng-template>
    <ng-template #headerTemplate="" let-data="">
      <!--set the value to headerTemplate property-->
      <span class='head'><span class='name'>Name</span><span
class='city'>City</span></span>
    </ng-template>
  </ejs-combobox>
</div>
</div>
```

Footer template

The ComboBox has options to show a footer element at the bottom of the list items in the popup list. Here, you can place any custom element as a footer element using

[footerTemplate](#) property.

In the following sample, footer element displays the total number of list items present in the ComboBox.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, ComboBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template url path
  templateUrl: 'template.html'
})
export class AppComponent {
  constructor() {
  }
  // defined the array of data
  public data: Object[] = ['Badminton', 'Basketball', 'Cricket', 'Hockey',
'Golf'];
  // set placeholder text to ComboBox input element
  public text: string = 'Select a game';
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

TEMPLATE.HTML

```
<div id="wrapper">
  <div id='content' style="margin: 40px auto 0; width:250px;">
    <!-- specifies the template string for the ComboBox component-->
    <ejs-combobox id='comboelement' #samples [dataSource]='data'
[placeholder]='text' [footerTemplate]='footerTemplate'>
      <ng-template #footerTemplate="" let-data="">
        <!--set the value to footerTemplate property-->
        <span class='foot'> Total list item: 5</span>
      </ng-template>
    </ejs-combobox>
  </div>
</div>
```

No records template

The ComboBox is provided with support to custom design the popup list content when no data is found and no matches found on search with the help of [noRecordsTemplate](#) property.

In the following sample, popup list content displays the notification of no data available.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule,ComboBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component
  template: `<ejs-combobox id='comboelement' [dataSource]='data'
placeholder='Find a item'>
      <ng-template #noRecordsTemplate>
        <span class='norecord'> NO DATA AVAILABLE</span>
      </ng-template>
    </ejs-combobox>`
})
export class AppComponent {
  // defined the empty array data
  public data: string[] = [];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Action failure template

There is also an option to custom design the popup list content when the data fetch request fails at the remote server. This can be achieved using the

[actionFailureTemplate](#) property.

In the following sample, when the data fetch request fails, the ComboBox displays the notification.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component } from '@angular/core';
//import data manager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
@Component({
  imports: [
    FormsModule, ComboBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component
  template: `<ejs-combobox id='comboelement' [dataSource]='data'
[query]='query' [fields]='fields' placeholder='Find an employee'>
    <ng-template #actionFailureTemplate>
      <span class='action-failure'> Data fetch get
fails</span>
    </ng-template>
  </ejs-combobox>`
})
export class AppComponent {
  //bind the data manager instance to dataSource property
  public data: DataManager = new DataManager({
    // Here, use the wrong url to display the action failure template
    url: 'https://services.odata.org/V4/Northwind/Northwind.svcs/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  });
  //bind the Query instance to query property
  public query: Query = new
Query().from('Employees').select(['FirstName']).take(6);
  // maps the appropriate column to fields property
  public fields: Object = { text: 'FirstName', value: 'EmployeeID' };
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [How to achieve filtering](#)
- [How to group the data using header](#)
- [How to show the list items with icon](#)

Virtualization in ComboBox Component

ComboBox virtualization is a technique used to efficiently render extensive lists of items while minimizing the impact on performance. This method is particularly advantageous when dealing with large datasets because it ensures that only a fixed number of DOM (Document Object Model) elements are created. When scrolling through the list, existing DOM elements are reused to display relevant data instead of generating new elements for each item. This recycling process is managed internally.

During virtual scrolling, the data retrieved from the data source depends on the popup height and the calculation of the list item height. Enabling the [enableVirtualization](#) option in a ComboBox activates this virtualization technique.

When fetching data from the data source, the [actionBegin](#) event is triggered before data retrieval begins. Then, the [actionComplete](#) event is triggered once the data is successfully fetched.

Furthermore, Incremental Search is supported with virtualization in the Combobox component. When a key is typed, the focus is moved to the respective element in the open popup state. In the closed popup state, the popup opens, and focus is moved to the respective element in the popup list based on the typed key. The Incremental Search functionality is well-suited for scenarios involving remote data binding.

When the `enableVirtualization` property is enabled, the `skip` and `take` properties provided by the user within the Query class at the initial state or during the `actionBegin` or `actionComplete` events will not be considered, since it is internally managed and calculated based on certain dimensions with respect to the popup height.

Binding local data

The Combobox can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property. When using virtual scrolling, the list updates based on the scroll offset value, triggering a request to fetch more data from the server. As the data is being fetched, the `actionBegin` event occurs before the data retrieval starts. Once the data retrieval is successful, the `actionComplete` event is triggered, indicating that the data fetch process is complete.

In the following example, `id` column and `text` column from complex data have been mapped to the `value` field and `text` field, respectively.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component } from '@angular/core';
import { ComboBoxComponent, VirtualScroll } from '@syncfusion/ej2-angular-dropdowns';
ComboBoxComponent.Inject(VirtualScroll);
@Component({
  imports: [
```

```

FormsModule, ComboBoxModule
],
standalone: true,
selector: 'app-root',
// specifies the virtual-scroll url path
templateUrl: 'virtual-scroll.html'
}))
export class AppComponent {
// defined the array of data
public records: { [key: string]: Object }[] = [];
constructor() {
for (let i: number = 1; i <= 150; i++) {
const item: { [key: string]: Object } = {
id: 'id' + i,
text: `Item ${i}`,
};
this.records.push(item);
}
}
// maps the appropriate column to fields property
public fields: object = { text: 'text', value: 'id' };
// set the placeholder to AutoComplete input
public waterMark: string = 'e.g. Item 1';
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

TEMPLATE.HTML

```

<div id="wrapper">
  <div id='content' style="margin: 40px auto 0; width:250px;">
    <!-- specifies the virtualization for the ComboBox component-->
    <ejs-combobox id='combobox-virtualization' [dataSource]='records'
[fields]='fields' [enableVirtualization]='true' [allowFiltering]='false'
popupHeight='200px' [placeholder]='waterMark'>
    </ejs-combobox>
  </div>
</div>

```

Binding remote data

The Combobox supports retrieval of data from remote data services with the help of **DataManager** component. When using remote data, it initially fetches all the data from the server, triggering the **actionBegin** and **actionComplete** events, and then stores the data locally. During virtual scrolling, additional data is retrieved from the locally stored data, triggering the **actionBegin** and **actionComplete** events at that time as well.

The following sample displays the OrderId from the **Orders** Data Service.

APP.COMPONENT.TS


```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component } from '@angular/core';
import { ComboBoxComponent, VirtualScroll } from '@syncfusion/ej2-angular-dropdowns';
import { Query, DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
ComboBoxComponent.Inject(VirtualScroll);
@Component({
  imports: [
    FormsModule, ComboBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the virtual-scroll url path
  templateUrl: 'virtual-scroll.html'
})
export class AppComponent {
  // bind the DataManager instance to dataSource property
  public customerData: DataManager = new DataManager({
    url: 'https://services.syncfusion.com/angular/production/api/Orders',
    adaptor: new WebApiAdaptor,
    crossDomain: true
  });
  // maps the appropriate column to fields property
  public customerField: { [key: string]: string } = { text: 'OrderID',
value: 'OrderID' };
  // set the placeholder to AutoComplete input
  public waterMark: string = 'OrderId';
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

TEMPLATE.HTML

```
<div id="wrapper">
  <div id='content' style="margin: 40px auto 0; width:250px;">
    <!-- specifies the virtualization for the ComboBox component-->
    <ejs-combobox id='combobox-virtualization'
[dataSource]='customerData' [fields]='customerField'
[enableVirtualization]='true' [allowFiltering]='true' popupHeight='200px'
[placeholder]='waterMark'>
      </ejs-combobox>
    </div>
  </div>
```

Grouping

The Combobox component supports grouping with Virtualization. It allows you to organize elements into groups based on different categories. Each item in the list can be classified using the [groupBy](#) field in the data table. After grouping, virtualization works similarly to local data binding, providing a seamless user experience. When the data source is bound to remote data, an initial request is made to retrieve all data for the purpose of grouping. Subsequently, the grouped data works in the same way as local data binding virtualization, enhancing performance and responsiveness.

The following sample shows the example for Grouping with Virtualization.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component } from '@angular/core';
import { ComboBoxComponent, VirtualScroll } from '@syncfusion/ej2-angular-
dropdowns';
ComboBoxComponent.Inject(VirtualScroll);
@Component({
  imports: [
    FormsModule, ComboBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the virtual-scroll url path
  templateUrl: 'virtual-scroll.html'
})
export class AppComponent {
  // defined the array of data
  public records: { [key: string]: Object }[] = [];
  constructor() {
    for (let i = 1; i <= 150; i++) {
      let id = 'id' + i;
      let text = `Item ${i}`;
      let group = 'Group A';
      // Generate a random number between 1 and 4 to determine the
group
      const randomGroup = Math.floor(Math.random() * 4) + 1;
      switch (randomGroup) {
        case 1:
          group = 'Group A';
          break;
        case 2:
          group = 'Group B';
          break;
        case 3:
          group = 'Group C';
          break;
        case 4:
          group = 'Group D';
          break;
        default:
          break;
      }
    }
  }
}
```

```

        this.records.push({id, text, group});
    }
}
// maps the appropriate column to fields property
public fields: object = { groupBy: 'group', text: 'text', value: 'id' };
// set the placeholder to AutoComplete input
public waterMark: string = 'e.g. Item 1';
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

TEMPLATE.HTML

```

<div id="wrapper">
    <div id='content' style="margin: 40px auto 0; width:250px;">
        <!-- specifies the virtualization for the ComboBox component-->
        <ejs-combobox id='combobox-virtualization' [dataSource]='records'
[fields]='fields' [enableVirtualization]='true' [allowFiltering]='true'
popupHeight='200px' [placeholder]='waterMark'>
            </ejs-combobox>
        </div>
    </div>

```

Filtering with Virtualization

The ComboBox component supports Filtering with Virtualization. The ComboBox includes a built-in feature that enables data filtering when the [allowFiltering](#) option is enabled. In the context of Virtual Scrolling, the filtering process operates in response to the typed characters. Specifically, the DropDownList sends a request to the server, utilizing the full data source, to achieve filtering. Before initiating the request, an action event is triggered. Upon successful retrieval of data from the server, an action complete event is triggered. The initial data is loaded when the popup is opened. Whether the filter list has a selection or not, the popup closes.

The following sample shows the example for Filtering with Virtualization.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component } from '@angular/core';
import { ComboBoxComponent, VirtualScroll } from '@syncfusion/ej2-angular-
dropdowns';
ComboBoxComponent.Inject(VirtualScroll);
@Component({
    imports: [
        FormsModule, ComboBoxModule
    ],
    standalone: true,

```

```

    selector: 'app-root',
    // specifies the virtual-scroll url path
    templateUrl: 'virtual-scroll.html'
  })
  export class AppComponent {
    // defined the array of data
    public records: { [key: string]: Object }[] = [];
    constructor() {
      for (let i: number = 1; i <= 150; i++) {
        const item: { [key: string]: Object } = {
          id: 'id' + i,
          text: `Item ${i}`,
        };
        this.records.push(item);
      }
    }
    // maps the appropriate column to fields property
    public fields: object = { text: 'text', value: 'id' };
    // set the placeholder to AutoComplete input
    public waterMark: string = 'e.g. Item 1';
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

TEMPLATE.HTML

```

<div id="wrapper">
  <div id='content' style="margin: 40px auto 0; width:250px;">
    <!-- specifies the virtualization for the ComboBox component-->
    <ejs-combobox id='combobox-virtualization' [dataSource]='records'
    [fields]='fields' [enableVirtualization]='true' [allowFiltering]='true'
    popupHeight='200px' [placeholder]='waterMark'>
    </ejs-combobox>
  </div>
</div>

```

Grouping in Angular Combo box component

The ComboBox supports wrapping nested elements into a group based on different categories. The category of each list item can be mapped through the [groupBy](#) field in the data table. The group header is displayed both as inline and fixed headers. The fixed group header content is updated dynamically on scrolling the popup list with its category value.

In the following sample, vegetables are grouped according on its category using `groupBy` field.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'

```

```
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component
  template: `<ejs-combobox #samples [dataSource]='data' [fields]='fields'
[placeholder]='text' [popupHeight]='height'></ejs-combobox>`
})
export class AppComponent {
  constructor() {
  }
  // defined the array of data
  public data: { [key: string]: Object }[] = [
    { Vegetable: 'Cabbage', Category: 'Leafy and Salad', Id: 'item1' },
    { Vegetable: 'Spinach', Category: 'Leafy and Salad', Id: 'item2' },
    { Vegetable: 'Wheat grass', Category: 'Leafy and Salad', Id: 'item3'
  },
    { Vegetable: 'Yarrow', Category: 'Leafy and Salad', Id: 'item4' },
    { Vegetable: 'Pumpkins', Category: 'Leafy and Salad', Id: 'item5' },
    { Vegetable: 'Chickpea', Category: 'Beans', Id: 'item6' },
    { Vegetable: 'Green bean', Category: 'Beans', Id: 'item7' },
    { Vegetable: 'Horse gram', Category: 'Beans', Id: 'item8' },
    { Vegetable: 'Garlic', Category: 'Bulb and Stem', Id: 'item9' },
    { Vegetable: 'Nopal', Category: 'Bulb and Stem', Id: 'item10' },
    { Vegetable: 'Onion', Category: 'Bulb and Stem', Id: 'item11' }];
  // maps the appropriate column to fields property
  public fields: Object = { groupBy: 'Category', text: 'Vegetable', value:
'Id' };
  // set the placeholder to the ComboBox input
  public text: string = "Select a vegetable";
  // Set the popup list height
  public height: string = '200px';
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customization

The grouping header is also provided with customization option. This allows custom designing using the [groupTemplate](#) property for both inline and fixed header.

See Also

- [Group Template support to ComboBox.](#)

Filtering in Angular Combo box component

The ComboBox has built-in support to filter the data items when [allowFiltering](#) enabled. The filter operation starts as soon as you start typing characters in the component.

By making use of [filtering](#) event, you can filter required data and return the data to ComboBox via [Link to the Video](#) method. So that those filtered items get displayed in the popup.

To get started quickly with Grouping and Filtering in angular ComboBox component, you can check the video below.

The following sample illustrates how to query the data source and pass the data to the ComboBox through the `updateData` method in `filtering` event.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
import { FilteringEventArgs } from '@syncfusion/ej2-dropdowns';
import { EmitType } from '@syncfusion/ej2-base';
import { Query } from '@syncfusion/ej2-data';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component with change event
  template: `<ejs-combobox id='combobox' #samples [dataSource]='data' [fields]='fields' [placeholder]='text' [allowFiltering]='true' (filtering)='onFiltering($event)'></ejs-combobox>`,
})
export class AppComponent {
  constructor() {
    // defined the array of data
    public data: { [key: string]: Object }[] = [
      { Id: "s3", Country: "Alaska" },
      { Id: "s1", Country: "California" },
      { Id: "s2", Country: "Florida" },
      { Id: "s4", Country: "Georgia" }];
    // maps the appropriate column to fields property
    public fields: Object = { text: "Country", value: "Id" };
    // set the placeholder to the ComboBox input
    public text: string = "Select a country";
    //Bind the filter event
    public onFiltering: EmitType<FilteringEventArgs> = (e: FilteringEventArgs) => {
      let query = new Query();
      //frame the query based on search string with filter type.
      query = (e.text != "") ? query.where("Country", "startswith", e.text, true) : query;
      //pass the filter data source, filter query to updateData method.
```

```
e.updateData(this.data, query);
};
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Limit the minimum filter character

When filtering the list items, you can set the limit for character count to raise remote request and fetch filtered data on the ComboBox. This can be done by manual validation within the filter event handler.

In the following example, the remote request does not fetch the search data until the search key contains three characters.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
import { FilteringEventArgs } from '@syncfusion/ej2-dropdowns';
import { EmitType } from '@syncfusion/ej2-base';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component with change event
  template: `<ejs-combobox id='comboelement' #samples [dataSource]='data'
[query]='query' [fields]='fields' [placeholder]='text'
[allowFiltering]='true' [sortOrder]='sorting'
(filtering)='onFiltering($event)'></ejs-combobox>`
})
export class AppComponent {
  constructor() {
  }
  //bind the DataManager instance to dataSource property
  public data: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  });
  public query: Query = new
  Query().from('Customers').select(['ContactName', 'CustomerID']).take(6);
  // maps the appropriate column to fields property
  public fields: Object = { text: 'ContactName', value: 'CustomerID' };
```

```
// set the placeholder to the ComboBox input
public text: string = "Select a customer";
//sort the result items
public sorting: string = 'Ascending';
//Bind the filter event
public onFiltering: EmitType<FilteringEventArgs> = (e:
FilteringEventArgs) => {
    // load overall data when search key empty.
    if (e.text === '') {
        e.updateData(this.data);
    } else {
        // restrict the remote request until search key contains 3
        characters.
        if (e.text.length < 3) { return; }
        let query: Query = new
Query().from('Customers').select(['ContactName', 'CustomerID']);
        query = (e.text !== '') ? query.where('ContactName', 'startswith',
e.text, true) : query;
        e.updateData(this.data, query);
    }
};
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Change the filter type

While filtering, you can change the filter type to `contains`, `startsWith`, or `endsWith` for string type within the filter event handler.

In the following examples, data filtering is done with `endsWith` type.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
import { FilteringEventArgs } from '@syncfusion/ej2-dropdowns';
import { EmitType } from '@syncfusion/ej2-base';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component
```



```

    template: `<ejs-combobox id='comboelement' #samples [dataSource]='data'
[query]='query' [fields]='fields' [placeholder]='text' popupHeight='250px'
[sortOrder]='sorting' [allowFiltering]='true'
(filtering)='onFiltering($event)'></ejs-combobox>`
  })
  export class AppComponent {
    constructor() {
    }
    //bind the DataManager instance to dataSource property
    public data: DataManager = new DataManager({
      url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
      adaptor: new ODataV4Adaptor,
      crossDomain: true
    });
    public query: Query = new
    Query().from('Customers').select(['ContactName', 'CustomerID']).take(7);
    // maps the appropriate column to fields property
    public fields: Object = { text: 'ContactName', value: 'CustomerID' };
    // set the placeholder to the ComboBox input
    public text: string = "Select a customer";
    //sort the result items
    public sorting: string = 'Ascending';
    //Bind the filter event
    public onFiltering: EmitType<FilteringEventArgs> = (e:
    FilteringEventArgs) => {
      // load overall data when search key empty.
      if (e.text === '') {
        e.updateData(this.data);
      } else {
        let query: Query = new
        Query().from('Customers').select(['ContactName', 'CustomerID']);
        // change the type of filtering
        query = (e.text !== '') ? query.where('ContactName', 'endswith',
        e.text, true) : query;
        e.updateData(this.data, query);
      }
    };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Case sensitive filtering

Data items can be filtered either with or without case sensitivity using the DataManager. This can be done by passing the fourth optional parameter of the `where` clause.

The following example shows how to perform case-sensitive filter.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'

```

```
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
import { FilteringEventArgs } from '@syncfusion/ej2-dropdowns';
import { EmitType } from '@syncfusion/ej2-base';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component with change event
  template: `<ejs-combobox id='comboelement' #samples [dataSource]='data' [query]='query' [fields]='fields' [placeholder]='text' popupHeight='250px' [sortOrder]='sorting' [allowFiltering]='true' (filtering)='onFiltering($event)'></ejs-combobox>`
})
export class AppComponent {
  constructor() {
    //bind the DataManager instance to dataSource property
    public data: DataManager = new DataManager({
      url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
      adaptor: new ODataV4Adaptor,
      crossDomain: true
    });
    public query: Query = new
    Query().from('Customers').select(['ContactName', 'CustomerID']).take(7);
    // maps the appropriate column to fields property
    public fields: Object = { text: 'ContactName', value: 'CustomerID' };
    // set the placeholder to the ComboBox input
    public text: string = "Select a customer";
    //sort the result items
    public sorting: string = 'Ascending';
    //Bind the filter event
    public onFiltering: EmitType<FilteringEventArgs> = (e:
    FilteringEventArgs) => {
      // load overall data when search key empty.
      if (e.text === '') {
        e.updateData(this.data);
      } else {
        let query: Query = new
        Query().from('Customers').select(['ContactName', 'CustomerID']);
        //enable the case sensitive filtering by passing false to 4th
        parameter.
        query = (e.text !== '') ? query.where('ContactName', 'startswith',
        e.text, false) : query;
        e.updateData(this.data, query);
      }
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Diacritics Filtering

The ComboBox supports diacritics filtering which will ignore the [diacritics](#) and makes it easier to filter the results in international characters lists when the [ignoreAccent](#) is enabled.

In the following sample, data with diacritics are bound as dataSource for ComboBox.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the DropDownList component with
  // change event
  template: `<ejs-combobox id='diacritics' [dataSource]='data'
[allowFiltering]='true' [ignoreAccent]='true' placeholder='e.g. aero'>
    </ejs-combobox>`
})
export class AppComponent {
  constructor() {
  }
  // create local data
  public data: string[] = [
    'Aeróbics',
    'Aeróbics en Agua',
    'Aerografía',
    'Aeromodelaje',
    'Águilas',
    'Ajedrez',
    'Ala Delta',
    'Álbumes de Música',
    'Alusivos',
    'Análisis de Escritura a Mano'];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [How to achieve autofill while filtering](#)
- [How to group the data using header](#)

Localization in Angular Combo box component

The Localization library allows you to localize static text content of the

[noRecordsTemplate](#) and [actionFailureTemplate](#) properties according to the culture currently assigned to the ComboBox.

| Locale key | en-US (default) |

|-----|-----|

| noRecordsTemplate | No records found |

| actionFailureTemplate | The request failed |

Loading translations

To load translation object to your application, use load function of the **L10n** class.

In the following sample, French culture is set to the ComboBox and no data is loaded. Hence, the [noRecordsTemplate](#) property displays its text in French culture initially, and if the sample is run offline, the [actionFailureTemplate](#) property displays its text appropriately.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit } from '@angular/core';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { L10n } from '@syncfusion/ej2-base';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component
  template: `<ejs-combobox id='comboelement' #samples [dataSource]='data'
[query]='query' [fields]='fields' [placeholder]='text'
[locale]='locale'></ejs-combobox>`
})
export class AppComponent implements OnInit {
  constructor() {
  }
  //set the placeholder text in french to ComboBox input
  public text: string = "Sélectionnez un élément";
  // bind remotedata to showcase actionFailureTemplate in offline
  public data: DataManager = new DataManager({
    url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ODataV4Adaptor,
```

```

        crossDomain: true
    });
    // map appropriate column
    public fields: Object = { text: 'ContactName', value: 'CustomerID' };
    // take 0 item to showcase noRecordsTemplate property
    public query: Query = new Query().select(['ContactName',
    'CustomerID']).take(0);
    //set culture to ComboBox component
    public locale: string = 'fr-BE';
    ngOnInit(): void {
        L10n.load({
            'fr-BE': {
                'dropdowns': {
                    'noRecordsTemplate': "Aucun enregistrement trouvé",
                    'actionFailureTemplate': "Modèle d'échec d'action"
                }
            }
        });
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Accessibility](#)
- [How to bind the data to the combobox](#)

Style in Angular Combo box component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of wrapper element

Use the following CSS to customize the appearance of wrapper element.

```

`css
.e-ddl.e-input-group.e-control-wrapper .e-input {
font-size: 20px;
font-family: emoji;
color: #ab3243;
background: #32a5ab;
}
`

```

Customizing the dropdown icon's color

Use the following CSS to customize the dropdown icon's color.

```
`css
.e-ddl.e-input-group .e-input-group-icon,.e-ddl.e-input-group.e-control-wrapper .e-input-group-
icon:hover {
color: #bb233d;
font-size: 13px;
}
`
```

Customizing the focus color

Use the following CSS to customize the focusing color of input element.

```
`css
.e-ddl.e-input-group.e-control-wrapper.e-input-focus::before, .e-ddl.e-input-group.e-control-wrapper.e-
input-focus::after {
background: #c000ff;
}
`
```

Customizing the outline theme's focus color

Use the following CSS to customize the focusing color of outline theme.

```
`css
.e-outline.e-input-group.e-input-focus:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-
disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-input-focus.e-control-wrapper:hover:not(.e-
success):not(.e-warning):not(.e-error):not(.e-disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-
input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled),.e-outline.e-input-group.e-
control-wrapper.e-input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled) {
border-color: #b1bd15;
box-shadow: inset 1px 1px #b1bd15, inset -1px 0 #b1bd15, inset 0 -1px #b1bd15;
}
`
```

Customizing the disabled component's text color

Use the following CSS to customize the text color when the component is disabled.

```
`css
.e-input-group.e-control-wrapper .e-input[disabled] {
-webkit-text-fill-color: #0d9133;
}
`
```

Customizing the float label element's focusing color

Use the following CSS to customize the focusing color of float label element.

```
`css
.e-float-input.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-control-
wrapper.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-input-group:not(.e-
float-icon-left) .e-float-line::after,.e-float-input.e-control-wrapper.e-input-group:not(.e-float-icon-left)
.e-float-line::after {
background-color: #2319b8;
}

.e-ddl.e-input-group.e-control-wrapper.e-float-input.e-input-focus .e-float-text.e-label-top, .e-float-
input.e-control-wrapper:not(.e-error).e-input-focus input ~ label.e-float-text {
color: #2319b8;
}
`
```

Customizing the color of the placeholder text

Use the following CSS to customize the text color of placeholder.

```
`css
.e-ddl.e-input-group input.e-input::placeholder {
color: red;
}
`
```

Customizing the text selection color

Use the following CSS to customize the selection color of text and background.

```
`css
.e-ddl.e-input-group input.e-input::selection {
color: red;
background: yellow;
}
`
```

Customizing the background color of focus, hover, and active item's

Use the following CSS to customize the background color of focus, hover and active item's.

```
`css
.e-dropdownbase .e-list-item.e-item-focus, .e-dropdownbase .e-list-item.e-active, .e-dropdownbase .e-
list-item.e-active.e-hover, .e-dropdownbase .e-list-item.e-hover {
background-color: #1f9c99;
color: #2319b8;
}
```

```
}
`
```

Customizing the appearance of pop-up element

Use the following CSS to customize the appearance of popup element.

```
`css
.e-dropdownbase .e-list-item, .e-dropdownbase .e-list-item.e-item-focus {
background-color: #29c2b8;
color: #207cd9;
font-family: emoji;
min-height: 29px;
}
`
```

Adding mandatory asterisk to placeholder and float label

You can add a mandatory asterisk(*) to placeholder and float label using `.e-input-group.e-control-wrapper.e-float-input .e-float-text::after` class.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component with
  // dataSource
  template: `<ejs-combobox id='comboelement' [dataSource]='data'
placeholder = 'Select a game' floatLabelType="Auto"></ejs-combobox>`
})
export class AppComponent {
  constructor() {
  }
  // defined the array of data
  public data: string[] = ['Cricket', 'Football', 'Rugby', 'Snooker',
'Tennis'];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
```



```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Accessibility in Angular Combo box component

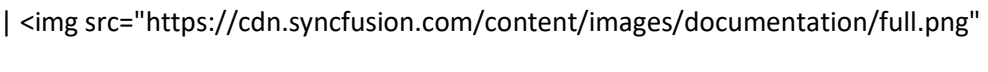
The ComboBox component has been designed, keeping in mind the WAI-ARIA specifications, and applies the WAI-ARIA roles, states, and properties along with keyboard support. This component is characterized by complete keyboard interaction support and ARIA accessibility support that makes it easy for people who use assistive technologies (AT) or those who completely rely on keyboard navigation.

The ComboBox component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the ComboBox component is outlined below.

| Accessibility Criteria | Compatibility |

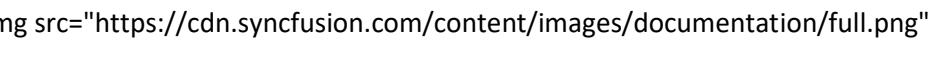
| -- | -- |

| [WCAG 2.2 Support](#) |  alt="Yes" > |

| [Section 508 Support](#) |  alt="Yes" > |

| [Screen Reader Support](#) |  alt="Yes" > |

| [Right-To-Left Support](#) |  alt="Yes" > |

| [Color Contrast](#) |  alt="Yes" > |

| [Mobile Device Support](#) |  alt="Yes" > |

| [Keyboard Navigation Support](#) |  alt="Yes" > |

| [Accessibility Checker Validation](#) |  alt="Yes" > |

| [Axe-core Accessibility Validation](#) |  alt="Yes" > |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The ComboBox component uses the **combobox** role, and each list item has an **option** role. The following **ARIA attributes** denote the ComboBox state.

| Properties | Functionalities |

| --- | --- |

| aria-haspopup | Indicates whether the ComboBox input element has a popup list or not. |

| aria-expanded | Indicates whether the popup list has expanded or not. |

| aria-selected | Indicates the selected option. |

| aria-readonly | Indicates the readonly state of the ComboBox element. |

| aria-disabled | Indicates whether the ComboBox component is in a disabled state or not. |

| aria-activedescendent | This attribute holds the ID of the active list item to focus its descendant child element. |

| aria-owns | This attribute contains the ID of the popup list to indicate popup as a child element. |

| aria-autocomplete | This attribute contains the 'both' to a list of options shows and the currently selected suggestion also shows inline. |

Keyboard interaction

You can use the following key shortcuts to access the ComboBox without interruptions.

| Keyboard shortcuts | Actions |

| --- | --- |

| Arrow Down | Selects the first item in the ComboBox when no item selected. Otherwise, selects the item next to the currently selected item. |

| Arrow Up | Selects the item previous to the currently selected one. |

| Page Down | Scrolls down to the next page and selects the first item when popup list opens. |

| Page Up | Scrolls up to the previous page and selects the first item when popup list opens. |

| Enter | Selects the focused item and popup list closes when it is in open state. |

| Tab | Focuses on the next TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the component. |

| Shift + tab | Focuses on the previous TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the component. |

| Alt + Down | Open the popup list |

| Alt + Up | Closes the popup list |

| Esc(Escape) | Closes the popup list when it is in an open state and the currently selected item remains the same. |

| Home | Cursor moves to before of first character in input |

| End | Cursor moves to next of last character in input |

In the following sample, alt+t keys are used to focus the ComboBox component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, HostListener, ViewChild } from '@angular/core';
import { ComboBoxComponent } from '@syncfusion/ej2-angular-dropdowns';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template string for the ComboBox component with change event
  template: `<ejs-combobox id='comboelement' #samples [dataSource]='data' [placeholder]='text' [popupHeight]='height'></ejs-combobox>`
})
export class AppComponent {
  @ViewChild('samples')
  public sports?: ComboBoxComponent;
  constructor() {
  }
  // defined the array of data
  public data: string[] = ['Badminton', 'Basketball', 'Cricket', 'Football', 'Golf', 'Hockey', 'Rugby', 'Snooker', 'Tennis'];
  // set placeholder to ComboBox input element
  public text: string = "Select a game";
  // set the popup list height
  public height: string = '200px';
  @HostListener('document:keyup', ['$event'])
  handleKeyboardEvent(event: KeyboardEvent) {
    if (event.altKey && event.keyCode === 84 /* t */) {
      // press alt+t to focus the control.
      this.sports!.focusIn();
    }
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Ensuring accessibility

The ComboBox component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the ComboBox component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the ComboBox component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Form support in Angular Combo box component

The ComboBox supports both the reactive and template-driven form-building technologies.

Template-Driven Forms

The template-driven forms uses the `ng` directives in view to handle the forms controls.

To enable the template-driven, import the FormsModule into corresponding app component.

For more details about template-driven Forms refer to: <https://angular.io/guide/forms#template-driven-forms>.

Mention the `name` attribute to ComboBox element which will be used to identify the

form element. To register an ComboBox element to ngForm, give the ngModel to it

so the FormsModule will automatically detect the ComboBox as a form element. After that, the ComboBox value will be selected based on the ngModel value.

The following example demonstrates how to achieve a two-way data binding.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { BrowserModule } from '@angular/platform-browser';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: 'form-support.html'
})
export class AppComponent {
  // defined the array of data
  public skillset: string[] = [
    'ASP.NET', 'ActionScript', 'Basic',
```

```

        'C++' , 'C#' , 'dBase' , 'Delphi' ,
        'ESPOL' , 'F#' , 'FoxPro' , 'Java' ,
        'J#' , 'Lisp' , 'Logo' , 'PHP'
    ];
    public placeholder: String = 'e.g: ActionScript';
    autoskillname: any;
    autosname: any;
    autosmail: any;
    constructor() {
    }
    skillForm = {
        skillname: null,
        sname: '',
        smail: ''
    };
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Reactive Forms

The reactive forms uses the reactive model-driven technique to handle form data between component and view, due to that we also call it as the **model-driven** forms. It's listen the form data changes between App component and view also returns the valid states and values of form elements.

For more details about Reactive Forms refer: <https://angular.io/guide/reactive-forms>.

For the reactive forms you should import a **ReactiveFormsModule** into app module as well as the **FormGroup**, **FormControl** should be imported to app component. The **FormGroup** is used to declare **formGroupName** for the form and the **FormControl** is used to declare **formControlName** for form controls.

You can declare the **formControlName** to **ComboBox** as usual. then, you must create a value object to the **FormGroup** and each value will be the default value of the form control.

The following example demonstrates how to use the reactive forms.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit, Inject } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { BrowserModule } from '@angular/platform-browser';
@Component({
    imports: [
        FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
    ],

```

```
standalone: true,
  selector: 'app-root',
  templateUrl: 'reactive-form.html'
}))
export class AppComponent {
  // defined the array of data
  public skillset: string[] = [
    'ASP.NET', 'ActionScript', 'Basic',
    'C++', 'C#', 'dBase', 'Delphi',
    'ESPOL', 'F#', 'FoxPro', 'Java',
    'J#', 'Lisp', 'Logo', 'PHP'
  ];
  public placeholder: String = 'e.g: ActionScript';
  skillForm?: FormGroup | any;
  fb: FormBuilder;
  constructor(@Inject(FormBuilder) private builder: FormBuilder) {
    this.fb = builder;
    this.createForm();
  }
  createForm() {
    this.skillForm = this.fb.group({
      skillname: ['', Validators.required],
      sname: ['', Validators.required],
      smail: ['', Validators.required]
    });
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

How To

Autofill in Angular Combo box component

The ComboBox supports the **autofill** behaviour with the help of [autofill](#) property. Whenever you change the input value, the ComboBox will autocomplete your data by matching the typed character. Suppose, if no matches found then, comboBox doesn't suggest any item.

The following examples, showcase that how to work autofill with ComboBox.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, ComboBoxModule, ButtonModule
  ],
```

```
standalone: true,
  selector: 'app-root',
  // specifies the template string for enable the autofill in ComboBox
  component
  template: `<ejs-combobox id='comboelement' #samples [dataSource]='data'
[autofill]='true' [placeholder]='text'></ejs-combobox>`
})
export class AppComponent {
  constructor() {
  }
  // defined the array of data
  public data: string[] = ['Badminton', 'Basketball', 'Cricket', 'Golf',
'Hockey', 'Rugby'];
  // set placeholder text to ComboBox input element
  public text: string = 'Select a game';
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Cascading in Angular Combo box component

The cascading ComboBox is a series of ComboBox, where the value of one ComboBox depends upon another's value. This can be configured by using the [change](#) event of the parent ComboBox. Within that change event handler, data has to be loaded to the child ComboBox based on the selected value of the parent ComboBox.

The following example, shows the cascade behavior of country, state, and city ComboBox. Here, the `dataBind` method is used to reflect the property changes immediately to the ComboBox.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, ViewChild } from '@angular/core';
import { ComboBoxComponent } from '@syncfusion/ej2-angular-dropdowns';
import { Query, DataManager } from '@syncfusion/ej2-data';
@Component({
  imports: [
    FormsModule, ComboBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  // specifies the template path for the ComboBox component
  templateUrl: `cascading.html`
})
export class AppComponent {
  constructor() {
  }
  //define the country ComboBox data
```

```

public countryData: { [key: string]: Object }[] = [
    { CountryName: 'Australia', CountryId: '2' },
    { CountryName: 'United States', CountryId: '1' }
];
//define the state ComboBox data
public stateData: { [key: string]: Object }[] = [
    { StateName: 'New York', CountryId: '1', StateId: '101' },
    { StateName: 'Virginia ', CountryId: '1', StateId: '102' },
    { StateName: 'Tasmania ', CountryId: '2', StateId: '105' }
];
//define the city ComboBox data
public cityData: { [key: string]: Object }[] = [
    { CityName: 'Albany', StateId: '101', CityId: 201 },
    { CityName: 'Beacon ', StateId: '101', CityId: 202 },
    { CityName: 'Emporia', StateId: '102', CityId: 206 },
    { CityName: 'Hampton ', StateId: '102', CityId: 205 },
    { CityName: 'Hobart', StateId: '105', CityId: 213 },
    { CityName: 'Launceston ', StateId: '105', CityId: 214 }
];
// maps the appropriate column to fields property for country ComboBox
public countryFields: Object = { text: 'CountryName', value: 'CountryId'
};
// maps the appropriate column to fields property for state ComboBox
public stateFields: Object = { text: 'StateName', value: 'StateId' };
// maps the appropriate column to fields property for city ComboBox
public cityFields: Object = { text: 'CityName', value: 'CityId' };
//set the placeholder to country ComboBox input
public countryWatermark: string = "Select a country";
//set the placeholder to state ComboBox input
public stateWatermark: string = "Select a state";
//set the placeholder to city ComboBox input
public cityWatermark: string = "Select a city";
@ViewChild('country')
// create object for country comboBox
public countryObj?: ComboBoxComponent | any;
@ViewChild('state')
// create object for state comboBox
public stateObj: ComboBoxComponent | any;
@ViewChild('city')
// create object for city comboBox
public cityObj?: ComboBoxComponent | any;
public countryChange(): void {
    let tempQuery: Query = new Query().where('CountryId', 'equal',
this.countryObj.value);
    //Query the data source based on country ComboBox selected value
    this.stateObj.query = tempQuery;
    // enable the state ComboBox
    this.stateObj.enabled = true;
    //clear the existing selection.
    this.stateObj.text = null;
    // bind the property changes to state ComboBox
    this.stateObj.dataBind();
    //clear the existing selection in city ComboBox
    this.cityObj.text = null;
    //disabe the city ComboBox
    this.cityObj.enabled = false;
    //bind the property cahnges to City ComboBox

```



```

        this.cityObj.dataBind();
    }
    public stateChange(): void {
        // Query the data source based on state ComboBox selected value
        this.cityObj.query = new Query().where('StateId', 'equal',
this.stateObj.value);
        // enable the city ComboBox
        this.cityObj.enabled = true;
        //clear the existing selection
        this.cityObj.text = null;
        // bind the property change to city ComboBox
        this.cityObj.dataBind();
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

CASCADING.HTML

```

<div id="wrapper" style='margin-top: 40px'>
  <br>
  <ejs-combobox #country id="country" [dataSource]="countryData"
[fields]="countryFields" (change)="countryChange()"
[placeholder]="countryWatermark"></ejs-combobox>
  <div class="padding-top">
    <ejs-combobox #state id="state" [dataSource]="stateData"
[fields]="stateFields" (change)="stateChange()"
[placeholder]="stateWatermark" [enabled]="false"></ejs-combobox>
  </div>
  <div class="padding-top">
    <ejs-combobox #city id="city" [dataSource]="cityData"
[fields]="cityFields" [placeholder]="cityWatermark" [enabled]="false"></ejs-
combobox>
  </div>
</div>

```

Icons support in Angular Combo box component

You can render **icons** to the list items by mapping the [iconCss](#) fields. This [iconCss](#) field create a span in the list item with mapped class name to allow styling as per your need.

In the following sample, icon classes are mapped with [iconCss](#) field.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ComboBoxModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component } from '@angular/core';
@Component({

```

```
imports: [
    FormsModule, ComboBoxModule
],
standalone: true,
selector: 'app-root',
// specifies the template string for the ComboBox component.
template: `<ejs-combobox id='comboelement' #samples [dataSource]='data'
[fields]='fields' [placeholder]='text'></ejs-combobox>`
})
export class AppComponent {
    constructor() {
    }
    // defined the array of data
    public data: { [key: string]: Object }[] = [
        { Class: 'asc-sort', Type: 'Sort A to Z', Id: '1' },
        { Class: 'dsc-sort', Type: 'Sort Z to A ', Id: '2' },
        { Class: 'filter', Type: 'Filter', Id: '3' },
        { Class: 'clear', Type: 'Clear', Id: '4' }];
    // map the icon column to iconCSS field.
    public fields: Object = { text: 'Type', iconCss: 'Class', value: 'Id' };
    //set the placeholder to ComboBox input
    public text: string = 'Select a format';
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Ej1 api migration in Angular Combo box component

This article describes the API migration process of ComboBox component from Essential JS 1 to Essential JS 2.

DataBinding

<!-- markdownlint-disable MD010 -->

Behavior	API in Essential JS 1	API in Essential JS 2
---	---	---
Default	Property: <code>dataSource</code> <code><input id="dropdown1" ej-combobox [dataSource]="data" /></code> Property: <code>dataSource</code> <code><ejs-combobox [dataSource]="dataSource"></ejs-combobox></code>	
Fields for mapping	Property: <code>fields</code> <code><input id="dropdown1" ej-combobox [fields]="fieldsvalues"/></code> Property: <code>fields</code> <code><ejs-combobox [fields]="fields"></ejs-combobox></code>	
Query	Property: <code>query</code> <code><input id="dropdown1" ej-combobox [query]="query"/></code> Property: <code>query</code> <code><ejs-combobox [query]="query"></ejs-combobox></code>	

| **Begin event** | **Event:** *actionBegin*
(actionBegin)="actionBegin(\$event)"/> | **Event:** *actionBegin*
(actionBegin)="actionBegin(\$event)"></ej-combobox> |

| **Complete event** | **Event:** *actionComplete*
(actionComplete)="actionComplete(\$event)"/> | **Event:** *actionComplete*
(actionComplete)="actionComplete(\$event)"></ej-combobox> |

| **Failure event** | **Event:** *actionFailure*
(actionFailure)="actionFailure(\$event)"/> | **Event:** *actionFailure*
(actionFailure)="actionFailure(\$event)"></ej-combobox> |

Filtering

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Default** | **Property:** *allowFiltering*
[allowFiltering]="allowFiltering"/> | **Property:** *allowFiltering*
[allowFiltering]="true"></ej-combobox> |

| **No records template** | **Property:** *noRecordsTemplate*
[noRecordsTemplate]="noRecordsTemplate"/> | **Property:** *noRecordsTemplate*
[noRecordsTemplate]="template"></ej-combobox> |

| **Ignore casing and diacritics** | **Not Applicable** | **Property:** *ignoreAccent*
[ignoreAccent] = "true"/> |

| **Custom value addition** | **Property:** *allowCustom*
[allowCustom]="allowCustom"/> | <https://ej2.syncfusion.com/angular/demos/#/material/combobox/custom-value> |

| **Search event** | **Event:** *filtering*
"onFiltering(\$event)"/> | **Event:** *filtering*
(filtering) = "onFiltering(\$event)"/> |

Template

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Default** | **Property:** *itemTemplate*
[itemTemplate]="itemTemplate"/> | **Property:** *itemTemplate*
[itemTemplate]="itemTemplate"></ej-combobox> |

| **Group Template** | **Property:** *groupTemplate*
[groupTemplate]="groupTemplate"/> | **Property:** *groupTemplate*
[groupTemplate]="groupTemplate"></ej-combobox> |

| **ValueTemplate** | **Not Applicable** | **Property:** *valueTemplate*
<ejs-combobox [valueTemplate] = "valueTemplate"/> |

| **Header Template** | **Property:** *headerTemplate*
<input id="dropdown1" ej-combobox [headerTemplate]="headerTemplate"/> | **Property:** *headerTemplate*
<ejs-combobox [headerTemplate]="headerTemplate"></ejs-combobox> |

| **FooterTemplate** | **Property:** *footerTemplate*
<input id="dropdown1" ej-combobox [footerTemplate]="footerTemplate"/> | **Property:** *footerTemplate*
<ejs-combobox [footerTemplate]="footerTemplate"></ejs-combobox> |

| **No records Template** | **Property:** *noRecordsTemplate*
<input id="dropdown1" ej-combobox [noRecordsTemplate]="noRecordsTemplate"/> | **Property:** *noRecordsTemplate*
<ejs-combobox [noRecordsTemplate]="noRecordsTemplate"></ejs-combobox> |

| **Auto fill** | **Property:** *autoFill*
<input id="dropdown1" ej-combobox [autoFill]="autoFill"/> | **Property:** *autoFill*
<ejs-combobox [autoFill]="true"></ejs-combobox> |

| **Action failure Template** | **Property:** *actionFailureTemplate*
<input id="dropdown1" ej-combobox [actionFailureTemplate]="actionFailureTemplate"/> | **Property:** *actionFailureTemplate*
<ejs-combobox [actionFailureTemplate]="actionFailureTemplate"></ejs-combobox> |

Applying CSS

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Default** | **Property:** *cssClass*
<input id="dropdown1" ej-combobox [cssClass]="cssClass"/> | **Property:** *cssClass*
<ejs-combobox [cssClass]="customclass"></ejs-combobox> |

| **width** | **Property:** *width*
<input id="dropdown1" ej-combobox [width]="width"/> | **Property:** *width*
<ejs-combobox [width]="200px"></ejs-combobox> |

Grouping

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Default** | **Property:** *fields*
<input id="dropdown1" ej-combobox [fields]="fields"/> | **Property:** *fields*
<ejs-combobox [fields]="field"></ejs-combobox> |

Accessibility

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Globalization** | **Property:** *locale*
<input id="dropdown1" ej-combobox [locale]="locale"/> |
Property: *locale*
<ejs-combobox [locale]="locale"/> |

| **Rtl support** | **Property:** *enableRtl*
<input id="dropdown1" ej-combobox
[enableRtl]="enableRtl"/> | **Property:** *enableRtl*
<ejs-combobox [enableRtl]="true"/> |

Placeholder

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Watermark text** | **Property:** *placeholder*
<input id="dropdown1" ej-combobox
[placeholder]="placeholder"/> |
Property: *placeholder*
<ejs-combobox
[placeholder]="select"/> |

| **Floating of watermark text** | **Not applicable** | **Property:** *floatLabelType*
<ejs-combobox
[floatLabelType]="floatLabelType"/> |

Miscellaneous

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Enable/disable** | **Property:** *enabled*
<input id="dropdown1" ej-combobox
[enabled]="enabled"/> | **Property:** *enabled*
<ejs-combobox [enabled]="true"/> |

| **Read only** | **Property:** *readOnly*
<input id="dropdown1" ej-combobox
[readOnly]="readOnly"/> | **Property:** *readOnly*
<ejs-combobox [readOnly]="true"/> |

| **Addition of Html attributes** | **Property:** *htmlAttributes*
<input id="dropdown1" ej-combobox
[htmlAttributes]="htmlAttributes"/> | **Property:** *htmlAttributes*
<ejs-combobox
[htmlAttributes]="htmlAttributes"/> |

Sorting

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Order of sorting** | **Property:** *sortOrder*
<input id="dropdown1" ej-combobox
[sortOrder]="sortOrder"/> | **Property:** *sortOrder*
<ejs-combobox [sortOrder]="sortOrder"/> |

Selection

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Selecting particular index** | **Property:** *index*
<input id="dropdown1" ej-combobox
[index]="index"/> | **Property:** *index*
<ejs-combobox [index]="index"/> |

| **Selecting particular value** | **Property:** *value*
Property: *value*
<ejs-combobox [value]="data"/> |

| **Selecting particular text** | **Property:** *text*
Property: *text*
<ejs-combobox [text]="data"/> |

| **Getting data by using value** | **Method:** *getItemDataByValue*
Method: *getDataByValue*
<ejs-combobox #sample id="combobox"/>

@ViewChild('sample') public cmbObj: ComboBoxComponent;

 cmbObj.getDataByValue("data");

| **Select event** | **Event:** *select*
Event: *select*
<ejs-combobox (select)="onSelect(\$event)"/> |

Popup

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Popup height** | **Property:** *popupHeight*
Property: *popupheight*
<ejs-combobox [popupHeight]="300px"/> |

| **Popup width** | **Property:** *popupWidth*
Property: *popupWidth*
<ejs-combobox [popupWidth]="300px"/> |

| **Popup showing manually** | **Method:** *showPopup*
Method: *showPopup*
<ejs-combobox #sample id="combobox"/>

@ViewChild('sample') public cmbObj: ComboBoxComponent;

 cmbObj.showPopup(); |

| **Popup hiding manually** | **Method:** *hidePopup*
Method: *hidePopup*
<ejs-combobox #sample id="combobox"/>

@ViewChild('sample') public cmbObj: ComboBoxComponent;

 cmbObj.hidePopup(); |

| **Popup hide event** | **Event:** *close*
Event: *close*
<ejs-combobox (close)="onClose(\$event)"/> |

| **Popup shown event** | **Event:** *open*
Event: *open*
<ejs-combobox (open)="onopen(\$event)"/> |

Common

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| **Adding new item** | **Method :** *addItem*
Method: *addItem*
<ejs-combobox

```
#sample id="combobox"/><br/> <br/>@ViewChild('sample') public cmbObj:
ComboBoxComponent;<br/><br/> cmbObj.addItem({Id: 'id', Game: 'Golf'},2);|

| Focus out event | Not applicable | Event: blur<br/><ejs-combobox (blur)="onblur($event)"/> |

| Focus in event | Event: focus<br/><input id="dropdown1" ej-combobox
(focus)="focus($event)"/> | Event: focusIn<br/><ejs-combobox (focusIn)="onopen($event)"/> |

| Focus out | Method: focusOut<br/><input id="dropdown1" ej-combobox /> <br/>
<br/>$('#dropdown').ejComboBox("focusOut");| Method: focusOut<br/><ejs-combobox #sample
id="combobox"/><br/> <br/>@ViewChild('sample') public cmbObj:
ComboBoxComponent;<br/><br/> cmbObj.focusOut(); |

| Focus in | Method: focusIn<br/><input id="dropdown1" ej-combobox /> <br/>
<br/>$('#dropdown').ejComboBox("focusIn");| Method: focusIn<br/><ejs-combobox #sample
id="combobox"/><br/> <br/>@ViewChild('sample') public cmbObj:
ComboBoxComponent;<br/><br/> cmbObj.focusIn(); |

| Getting the data | Method : getItems<br/><input id="dropdown1" ej-combobox /> <br/>
<br/>$('#dropdown').ejComboBox("getItems");| Method: getItems<br/><ejs-combobox #sample
id="combobox"/><br/> <br/>@ViewChild('sample') public cmbObj:
ComboBoxComponent;<br/><br/> cmbObj.getItems();|

| Create event | Event: create<br/><input id="dropdown1" ej-combobox
(create)="create($event)"/> | Event: created<br/><ejs-combobox
(created)="created($event)"/> |

| Change event | Event: change<br/><input id="dropdown1" ej-combobox
(change)="change($event)"/> | Event: change<br/><ejs-combobox (change)="change($event)"/>
|

| Custom value event | Event: customValueSpecifier<br/><input id="dropdown1" ej-combobox
(customValueSpecifier)="customValueSpecifier($event)"/> | Event:
customValueSpecifier<br/><ejs-combobox
(customValueSpecifier)="customValueSpecifier($event)"/> |
```

ContextMenu

Getting started with Angular Context menu component

This section explains how to create a simple ContextMenu, and demonstrate the basic usage of the ContextMenu component in an Angular environment.

Dependencies

The list of dependencies required to use the Angular ContextMenu component in your application is given below:

```
`javascript
|-- @syncfusion/ej2-angular-navigations
|-- @syncfusion/ej2-angular-base
|-- @syncfusion/ej2-navigations
```

```
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-splitbuttons
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-buttons
`
```

Setup Angular environment

You can use [Angular CLI](#) to setup your Angular applications. To install Angular CLI use the following command.

```
npm install -g @angular/cli
```

Create an Angular application

Start a new Angular application using below Angular CLI command.

```
ng new my-app
cd my-app
```

Installing Syncfusion ContextMenu Package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(`>=20.2.36`) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-navigations](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-navigations --save
```


Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the ngcc package use the below.

Add [@syncfusion/ej2-angular-navigations@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-navigations@ngcc --save
`
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-navigations:"20.2.38-ngcc"
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding ContextMenu module

Import ContextMenu module into Angular application(`app.module.ts`) from the package

[@syncfusion/ej2-angular-navigations](#).

```
`javascript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Imported Syncfusion contextmenu module from navigations package
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations';
import { AppComponent } from './app.component';
@NgModule({
  imports: [ BrowserModule, ContextMenuModule ], // Registering EJ2 ContextMenu Module
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
`
```

Adding Syncfusion ContextMenu component

Modify the template in `app.component.ts` file with `ejs-contextmenu` to render the ContextMenu component and the option contains `menuItems` and `target` in which ContextMenu will be opened.

```
`javascript
import { Component } from '@angular/core';
```

```
import { MenuItemModel } from '@syncfusion/ej2-navigations';
@Component({
  selector: 'app-root',
  template: `<!--target element-->
<div id="target">Right click / Touch hold to open the ContextMenu</div>
<!-- To Render ContextMenu. -->
<ejs-contextmenu id='contextmenu' target='#target' [items]= 'menuItems'></ejs-contextmenu>`
})
export class AppComponent {
  public menuItems: MenuItemModel[] = [
    {
      text: 'Cut'
    },
    {
      text: 'Copy'
    },
    {
      text: 'Paste'
    }
  ];
}
```

Adding CSS reference

Add ContextMenu component's styles as given below in `style.css`.

```
`css
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
/ Context Menu target /
target {
  border: 1px dashed;
  height: 150px;
  padding: 10px;
  position: relative;
  text-align: justify;
  color: gray;
  user-select: none;
```

```
}
,
```

Running the application

Run the application in the browser using the following command:

```
,
```

```
ng serve
```

```
,
```

The following example shows a basic `ContextMenu` component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
import { MenuItemModel } from '@syncfusion/ej2-navigations';
@Component({
  imports: [

    ContextMenuModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!--target element-->
    <div id="target">Right click / Touch hold to open the
ContextMenu</div>
    <!-- To Render ContextMenu. -->
    <ejs-contextmenu id='contextmenu' target='#target' [items]=
'menuItems'></ejs-contextmenu>
    </div>`
})
export class AppComponent {
  public menuItems: MenuItemModel[] = [
    {
      text: 'Cut'
    },
    {
      text: 'Copy'
    },
    {
      text: 'Paste'
    }
  ];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Rendering items with Separator

The Separators are the horizontal lines used to separate the menu items. You cannot select the separators. You can enable separators to group the menu items using the [separator](#) property. Cut, Copy, and Paste menu items are grouped using [separator](#) property in the following sample.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
import { MenuItemModel } from '@syncfusion/ej2-navigations';
@Component({
  imports: [
    ContextMenuModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id="target">Right click / Touch hold to open the
ContextMenu</div>
    <ejs-contextmenu id='contextmenu' target='#target' [items]=
'menuItems'></ejs-contextmenu>
  </div>`
})
export class AppComponent {
  public menuItems: MenuItemModel[] = [
    {
      text: 'Cut'
    },
    {
      text: 'Copy'
    },
    {
      text: 'Paste'
    },
    {
      separator: true
    },
    {
      text: 'Font'
    },
    {
      text: 'Paragraph'
    }
  ];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

The [separator](#) property should not be given along with

the other fields in the [MenuItem](#).

Icons and navigation in Angular Context menu component

Icons

The ContextMenu item can have an icon/image in it to provide visual representation of the action. To place the icon on a menu item, set the [iconCss](#) property to e-icons with the required icon CSS. By default, the icon is positioned to the left side of the menu item. In the following sample, the icons for Cut, Copy and Paste menu items are added using the [iconCss](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
import { MenuItemModel } from '@syncfusion/ej2-navigations';
@Component({
  imports: [
    ContextMenuModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id="target">Right click / Touch hold to open the
    ContextMenu</div>
    <ejs-contextmenu id='contextmenu' target='#target'
    [items]='menuItems'></ejs-contextmenu>
    </div>`
})
export class AppComponent {
  public menuItems: MenuItemModel[] = [
    {
      text: 'Cut',
      iconCss: 'e-cm-icons e-cut'
    },
    {
      text: 'Copy',
      iconCss: 'e-cm-icons e-copy'
    },
    {
      text: 'Paste',
      iconCss: 'e-cm-icons e-paste',
    }
  ];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
```

```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Navigation

Navigation in ContextMenu is usage to navigate to the other web page when menu item is clicked. This can be achieved by providing link to the menu item using the [url](#) property.

In the following sample, Navigation URL for Flipkart, Amazon, and Snapdeal menu items are added using the `url` property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
import { MenuItemModel, MenuEventArgs } from '@syncfusion/ej2-navigations';
@Component({
  imports: [
    ContextMenuModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id="target">Right click / Touch hold to open the
ContextMenu</div>
    <ejs-contextmenu id='contextmenu' target='#target'
[items]='menuItems'
(beforeItemRender)='itemBeforeEvent($event)'></ejs-contextmenu>
</div>`,
})
export class AppComponent {
  public menuItems: MenuItemModel[] = [
    {
      text: 'Flipkart',
      iconCss: 'e-cart-icon e-link',
      url: 'https://www.google.co.in/search?source=hp&q=flipkart'
    },
    {
      text: 'Amazon',
      iconCss: 'e-cart-icon e-link',
      url: 'https://www.google.co.in/search?q=amazon'
    },
    {
      text: 'Snapdeal',
      iconCss: 'e-cart-icon e-link',
      url: 'https://www.google.co.in/search?q=snapdeal'
    }
  ];
  public itemBeforeEvent (args: MenuEventArgs) {
    args.element.getElementsByTagName('a')[0].setAttribute('target',
'_blank');
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

To open the links in new tab, set `target` attribute with the value `_blank` in the `beforeItemRender` event.

See Also

- [How to change menu items dynamically](#)

Templates in Angular Context menu component

Template

The ContextMenu items can be customized using the `beforeItemRender` property. The item render event triggers while rendering each menu item. The event argument will be used to identify the menu item and customized it based on the requirement. In the following sample, the menu item is rendered with keycode for specified action in ContextMenu using the template. Here, the keycode is specified for Save as, View page source, and Inspect in the right side corner of the menu items by adding span element in the `beforeItemRender` event.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
import { createElement } from '@syncfusion/ej2-base';
import { MenuEventArgs, MenuItemModel } from '@syncfusion/ej2-navigations';
@Component({
  imports: [
    ContextMenuModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!--target element-->
    <div id="target">Right click / Touch hold to open the
ContextMenu</div>
    <!-- To Render ContextMenu. -->
    <ejs-contextmenu id='contextmenu' #contextmenu target='#target'
[items]='menuItems' (beforeItemRender)="itemBeforeEvent($event)"></ejs-
contextmenu>
    </div>`
})
export class AppComponent {
  public menuItems: MenuItemModel[] = [
    {
      text: 'Save as...'
    }
  ]
}
```

```

    },
    {
      text: 'View page source'
    },
    {
      text: 'Inspect'
    }
  ]];
  public itemBeforeEvent (args: MenuEventArgs) {
    let shortCutSpan: HTMLElement = createElement('span');
    let text: string = args.item.text as string;
    let shortCutText: string = text === 'Save as...' ? 'Ctrl + S' : (text
    === 'View page source' ? 'Ctrl + U' : 'Ctrl + Shift + I');
    shortCutSpan.textContent = shortCutText;
    args.element.appendChild(shortCutSpan);
    shortCutSpan.setAttribute('class', 'shortcut');
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

To create span element, `createElement` utility function used from `ej2-base`.

Multilevel nesting

The Multiple level nesting supports in ContextMenu. It can be achieved by mapping the `items` property inside the parent `menuitems`. In the following sample, three level nesting of ContextMenu is provided.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
import { Component } from '@angular/core';
import { MenuItemModel } from '@syncfusion/ej2-navigations';
@Component({
  imports: [
    ContextMenuModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!--target element-->
    <div id="target">Right click / Touch hold to open the
ContextMenu</div>
    <!-- To Render ContextMenu. -->
    <ejs-contextmenu id='contextmenu' #contextmenu target='#target'
[items]='menuItems'></ejs-contextmenu>
    </div>`
})

```



```
export class AppComponent {
  public menuItems: MenuItemModel[] = [
    {
      text: 'Show All Bookmarks'
    },
    {
      text: 'Bookmarks Toolbar',
      items: [
        {
          text: 'Most Visited',
          items: [
            {
              text: 'Gmail'
            },
            {
              text: 'Google'
            }
          ]
        },
        {
          text: 'Recently Added'
        }
      ]
    }
  ]
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

To open sub menu items only on click, [showItemOnClick](#) property should be set as `true`.

See Also

- [Populate menu items with data source](#)

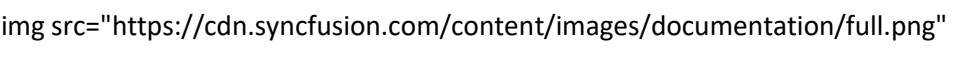
Accessibility in Angular Context menu component

The Context menu component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Context menu component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes" > |

| [Section 508](#) Support |  alt="Yes" > |

```

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| Accessibility Checker Validation |  |

| Axe-core Accessibility Validation |  |

<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

```

WAI-ARIA attributes

The Context menu component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Context menu component:

Attributes	Purpose
---	---
<code>role</code>	Indicates Context menu component popup as <code>menu</code> , and the popup items as <code>menuitem</code> .
<code>aria-haspopup</code>	Indicates the availability and type of interactive popup element.
<code>aria-expanded</code>	Indicates whether the subtree can be expanded or collapsed, as well as indicates whether its current state is expanded or collapsed.
<code>aria-label</code>	Indicates the menu item text.

Keyboard interaction

The Context menu component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Context menu component.

| **Press** | **To do this** |

| --- | --- |

| **Esc** | Closes the opened sub menu. |

| **Enter** | Selects the focused item. |

| **Up** | Navigates up or to the previous menu item. |

| **Down** | Navigates down or to the next menu item. |

| **Left** | Close the current sub menu and navigates to the parent menu. |

| **Right** | Navigates and open the next sub menu. |

Ensuring accessibility

The Context menu component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Context menu component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Context menu component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Style and appearance in Angular Context menu component

To modify the ContextMenu appearance, you need to override the default CSS of ContextMenu component. Please find the list of CSS classes and its corresponding section in ContextMenu component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

| CSS Class | Purpose of Class |

|-----|-----|

| .e-contextmenu-wrapper | To customize the context menu wrapper |

| .e-contextmenu-wrapper .e-menu-parent | To customize the context menu items |

| .e-contextmenu-wrapper ul .e-menu-item.e-selected .e-caret::before | To customize the context menu caret icon |

| .e-contextmenu-wrapper ul .e-menu-item .e-menu-icon::before | To customize the icons of the context menu |

How To

Populate menu items with data source in Angular Context menu component

To bind local data source to the ContextMenu, menu items are populated from data source and mapped to [items](#) property.

The below example demonstrates how to bind local data source to the ContextMenu.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
import { MenuEventArgs, MenuItemModel } from '@syncfusion/ej2-navigations';
import { Record, data } from './datasource';
@Component({
  imports: [

    ContextMenuModule,
    ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!--target element-->
    <div id="target">Right click / Touch hold to open the
ContextMenu</div>
    <!-- To Render ContextMenu. -->
    <ejs-contextmenu id='contextmenu' target='#target' [items]=
'items'
      (beforeItemRender)='itemRender($event)'></ejs-contextmenu>
    </div>`
})
export class AppComponent {
  public items: MenuItemModel[] = this.Items();
  public Items() {
    let record: Record;
    let menuItems: any[] = [];
    for (let i = 0; i < data.length; i++) {
      record = data[i] as Record;
      if (record.parentId) {
        if (!menuItems[record.parentId - 1].items) {
          menuItems[record.parentId - 1].items = []
        }
        menuItems[record.parentId - 1].items.push({ text: record.text });
      } else {
        menuItems.push({ text: record.text });
      }
    }
    return menuItems;
  }
  public itemRender(args: MenuEventArgs) {
    if (!args.item.text) {
      args.element.classList.add('e-separator');
    }
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Context menu item click in Angular Context menu component

This section explains about how to open a sub menu on Context Menu item click. This can be achieved by using `ShowItemOnClick` property of the Context Menu.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
import { MenuItemModel } from '@syncfusion/ej2-angular-navigations';
enableRipple(true);
@Component({
  imports: [
    ContextMenuModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!-- To Render Menu. -->
    <div id="target">Right click / Touch hold to open the
ContextMenu</div>
    <ejs-contextmenu #contextmenu target="#target"
[items]='menuItems' showItemOnClick="true" ></ejs-contextmenu>
    </div>`
})
export class AppComponent {
  public menuItems: MenuItemModel[] = [
    {
      text: 'Show All Bookmarks'
    },
    {
      text: 'Bookmarks Toolbar',
      items: [
        {
          text: 'Most Visited',
          items:[
            {
              text: 'Gmail'
            },
            {
              text: 'Google'
            }
          ]
        },
        {
          text: 'Recently Added'
        }
      ]
    }
  ]
}
```

```

    ]
  }
];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Open and close contextmenu in Angular Context menu component

ContextMenu can be opened and closed programmatically whenever required by using open and close methods.

In the following example, the ContextMenu is opened using the [open](#) method at the specified position using `top` and `left`. Also, ContextMenu is closed using [close](#) method on ContextMenu item click or document click.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
import { Component } from '@angular/core';
import { getInstance } from '@syncfusion/ej2-base';
import { MenuItemModel, ContextMenu } from '@syncfusion/ej2-navigations';
@Component({
  imports: [
    ContextMenuModule,
    ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!-- To Render ContextMenu. -->
    <ejs-contextmenu id='contextmenu' [items]= 'menuItems'></ejs-
contextmenu>
    <!-- To Render Button. -->
    <button ej2-button (click)="btnClick()">Open ContextMenu</button>
  </div>`
})
export class AppComponent {
  // Initialize action items.
  public menuItems: MenuItemModel[] = [
    {
      text: 'Cut'
    },
    {
      text: 'Copy'
    },
  ],
}

```

```

        {
            text: 'Paste'
        }
    ]];
    btnClick() {
        let contextmenuObj: ContextMenu =
        getInstance(document.getElementById("contextmenu_0") as HTMLElement,
        ContextMenu) as ContextMenu;
        contextmenuObj.open(40, 20);
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Change menu items dynamically in Angular Context menu component

The items visible in the ContextMenu can be changed dynamically based on the target in which you open the ContextMenu. To achieve this behavior, initialize ContextMenu with all items using [items](#) property and then based on the context you open hide/show required items using [hideItems](#)/[showItems](#) method in [beforeOpen](#) event.

In the following example, the datasource for Clipboard div is **Cut, Copy, Paste** and for the Editor div is **Add, Edit, Delete** is changed on [beforeOpen](#) event using [hideItems](#) and [showItems](#) method.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, ViewChild } from '@angular/core';
import { MenuItemModel, BeforeOpenCloseMenuEventArgs } from '@syncfusion/ej2-navigations';
import { ContextMenuComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
    imports: [

        ContextMenuModule
    ],
    standalone: true,
    selector: 'app-root',
    template: `<div class="e-section-control">
        <!--target element-->
        <div id="target">
            <div id='left' class='e-div'>Clipboard</div>
            <div id='right' class='e-div'>Editor</div>
        </div>
        <!-- To Render ContextMenu. -->
        <ejs-contextmenu #contextmenu target='#target .e-div' [items]=
        'menuItems' (beforeOpen)='beforeOpen($event)'></ejs-contextmenu>
        </div>`
    })

```

```

export class AppComponent {
  @ViewChild('contextmenu')
  public cmenu?: ContextMenuComponent;
  // Initialize menu items.
  public menuItems: MenuItemModel[] = [
    {
      text: 'Cut'
    },
    {
      text: 'Copy'
    },
    {
      text: 'Paste'
    },
    {
      text: 'Add'
    },
    {
      text: 'Edit'
    },
    {
      text: 'Delete'
    }
  ];
  public beforeOpen (args: BeforeOpenCloseMenuEventArgs) {
    // To hide/show items on right click.
    if ((args.event.target as HTMLElement).id === 'right') {
      (this.cmenu as ContextMenuComponent).hideItems(['Cut',
      'Copy', 'Paste']);
      (this.cmenu as ContextMenuComponent).showItems(['Add',
      'Edit', 'Delete']);
    } else if ((args.event.target as HTMLElement).id === 'left') {
      (this.cmenu as ContextMenuComponent).showItems(['Cut', 'Copy',
      'Paste']);
      (this.cmenu as
      ContextMenuComponent).hideItems(['Add', 'Edit', 'Delete']);
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Template in Angular Context menu component

Show table in sub ContextMenu

Menu items of the ContextMenu can be customized according to the requirement. The section explains about how to customize table template in sub menu item.

This can be achieved by appending table layout while `li` rendering by using `beforeItemRender` event.

APP.COMPONENT.TS


```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
import { ContextMenu, BeforeOpenCloseMenuEventArgs, MenuEventArgs,
MenuItemModel } from '@syncfusion/ej2-navigations';
import { createCheckBox } from '@syncfusion/ej2-buttons';
import { closest } from '@syncfusion/ej2-base';
@Component({
  imports: [

    ContextMenuModule,
    ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id="target">Right click / Touch hold to open the ContextMenu</div>
    <ejs-contextmenu id='contextmenu' [target]='target' [items]= 'menuItems'
    (beforeItemRender)='itemRender($event)'></ejs-contextmenu>
  </div>`
})
export class AppComponent {
  public target: string = '#target';
  public menuItems: MenuItemModel[] = [
    {
      text: 'Cut',
      iconCss: 'e-cm-icons e-cut'
    },
    {
      text: 'Copy',
      iconCss: 'e-cm-icons e-copy'
    },
    {
      text: 'Paste',
      iconCss: 'e-cm-icons e-paste'
    },
    {
      separator: true
    },
    {
      text: 'Link',
      iconCss: 'e-icons e-link'
    },
    {
      text: 'Table',
      iconCss: 'e-icons e-table',
      items: [
        {
          id: 'table'
        }
      ]
    }
  ]
};
public itemRender(args: MenuEventArgs) {
  if (args.item.id === 'table') {

```

```

        args.element.classList.add('bg-transparent');
        args.element.appendChild(this.createHeader());
        args.element.appendChild(this.createTable());
    }
}
public createHeader() {
    let header: HTMLElement = document.createElement('h4');
    header.textContent = 'Insert Table';
    return header;
}
public createTable() {
    let table: HTMLElement = document.createElement('table');
    for (let i: number = 0; i < 5; i++) {
        let row: HTMLElement = document.createElement('tr');
        table.appendChild(row);
        for (let j: number = 0; j < 6; j++) {
            let col: HTMLElement = document.createElement('td');
            row.appendChild(col);
            col.setAttribute('class', 'e-data');
        }
    }
    return table;
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Show UI components in ContextMenu

UI components can also be placed inside the each `li` element of ContextMenu.

In the following example, CheckBox component is placed inside each `li` element and this can be achieved by creating CheckBox component in `beforeItemRender` event and appending it into the `li` element.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
import { BeforeOpenCloseMenuEventArgs, MenuEventArgs, MenuItemModel } from '@syncfusion/ej2-navigations';
import { createCheckBox } from '@syncfusion/ej2-buttons';
import { closest, createElement } from '@syncfusion/ej2-base';
@Component({
    imports: [

        ContextMenuModule,
        ButtonModule
    ]
})

```

```

    ],
    standalone: true,
    selector: 'app-root',
    template: `<div class="e-section-control">
    <div id="target">Right click / Touch hold to open the ContextMenu</div>
    <ejs-contextmenu id='contextmenu' [target]='target' [items]= 'menuItems'
    (beforeItemRender)='itemRender($event)'
    (beforeClose)='beforeClose($event)'></ejs-contextmenu>
    </div>`
  })
  export class AppComponent {
    public target: string = '#target';
    public menuItems: MenuItemModel[] = [
      { text: 'Option 1' },
      { text: 'Option 2' },
      { text: 'Option 3' }
    ];
    public beforeClose(args: BeforeOpenCloseMenuEventArgs) {
      if ((args.event.target as Element).closest('.e-menu-item')) {
        args.cancel = true;
        let selectedElem: NodeList = args.element.querySelectorAll('.e-selected');
        for(let i:number=0; i < selectedElem.length; i++){
          let ele: Element = selectedElem[i] as Element;
          ele.classList.remove('e-selected');
        }
        let checkbox: HTMLElement = closest(args.event.target as Element,
        '.e-checkbox-wrapper') as HTMLElement;
        let frame: HTMLElement = checkbox.querySelector('.e-frame') as
        HTMLElement;
        if (checkbox && frame.classList.contains('e-check')) {
          frame.classList.remove('e-check');
        } else if (checkbox) {
          frame.classList.add('e-check');
        }
      }
    }
    public itemRender(args: MenuEventArgs) {
      let check: Element = createCheckBox(createElement, false, {
        label: args.item.text,
        checked: (args.item.text == 'Option 2') ? true : false
      });
      args.element.innerHTML = '';
      args.element.appendChild(check);
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Underline a character in the item text in Angular Context menu component

To underline a particular character in a text, it can be handled in `beforeItemRender` event by adding `<u>` tag in between the text and given as innerHTML in `li` rendering.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
import { MenuItemModel, MenuEventArgs } from '@syncfusion/ej2-navigations';
@Component({
  imports: [

    ContextMenuModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id="target">Right click / Touch hold to open the
ContextMenu</div>
    <ejs-contextmenu id='contextmenu' target='#target' [items]=
'menuItems' (beforeItemRender)= 'itemRender($event)'></ejs-contextmenu>
  </div>`
})
export class AppComponent {
  public menuItems: MenuItemModel[] = [
    {
      text: 'Cut'
    },
    {
      text: 'Copy'
    },
    {
      text: 'Paste'
    }
  ];
  public itemRender(args: MenuEventArgs) {
    if (args.item.text === "Copy") {
      args.element.innerHTML = '<u>C</u>opy';
    }
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Open a dialog on contextmenu item click in Angular Context menu component

This section explains about how to open a dialog on ContextMenu item click. This can be achieved by handling dialog open in `select` event of the ContextMenu.

In the following sample, Dialog will open while clicking **Save As...** item:

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, ViewChild } from '@angular/core';
import { MenuItemModel, MenuEventArgs } from '@syncfusion/ej2-navigations';
import { DialogComponent } from '@syncfusion/ej2-angular-popups';
@Component({
  imports: [

    DialogModule,
    ContextMenuModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id="target">Right click / Touch hold to open the ContextMenu</div>
    <ejs-dialog #dialog [buttons]='alertDlgButtons' [visible]='visible'
content='This file can be saved as PDF' width='200px' height='110px'
[position]='position'>
    </ejs-dialog>
    <ejs-contextmenu target="#target" [items]="data"
(select)="itemSelect($event)"></ejs-contextmenu></div>`
})
export class AppComponent {
  @ViewChild('dialog')
  public alertDialog?: DialogComponent;
  name = 'Angular';
  data= [
    {
      text: 'Back'
    },
    {
      text: 'Forward'
    },
    {
      text: 'Reload'
    },
    {
      separator: true
    },
    {
      text: 'Save As...'
    },
    {
      text: 'Print'
    },
    {
      text: 'Cast'
    }
  ];
  public visible: Boolean = false;
```

```

public position: any = {X: 100, Y: 100};
public alertDlgButtons: Object[] = [{
  buttonModel: {
    isPrimary: true,
    content: 'Submit',
    cssClass: 'e-flat',
  },
  click: function () {
    (this as any).hide();
  }
}];
public itemSelect(args: MenuEventArgs): void {
  if (args.item.text === "Save As...") {
    (this.alertDialog as DialogComponent).show();
  }
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Change animation settings in Angular Context menu component

To change the animation of the ContextMenu, [animationSettings](#) property is used.

The supported effects for ContextMenu are,

Effect	Functionality
-----	-----
None	Specifies the sub menu transform with no animation effect.
SlideDown	Specifies the sub menu transform with slide down effect.
ZoomIn	Specifies the sub menu transform with zoom in effect.
FadeIn	Specifies the sub menu transform with fade in effect.

The following sample illustrates how to open ContextMenu with **FadeIn** effect with the **duration** of **800ms**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
import { MenuItemModel } from '@syncfusion/ej2-navigations';
@Component({
  imports: [
    ContextMenuModule

```

```

    ],
    standalone: true,
    selector: 'app-root',
    template: `<div class="e-section-control">
      <!--target element-->
      <div id="target">Right click / Touch hold to open the
ContextMenu</div>
      <!-- To Render ContextMenu. -->
      <ejs-contextmenu id='contextmenu' #contextmenu target='#target'
[items]='menuItems' [animationSettings]='animation'></ejs-contextmenu>
      </div>`
  })
  export class AppComponent {
    public animation = {
      effect: 'FadeIn',
      duration: 800
    };
    public menuItems: MenuItemModel[] = [
      {
        text: 'Show All Bookmarks'
      },
      {
        text: 'Bookmarks Toolbar',
        items: [
          {
            text: 'Most Visited',
            items: [
              {
                text: 'Gmail'
              },
              {
                text: 'Google'
              }
            ]
          },
          {
            text: 'Recently Added'
          }
        ]
      }
    ];
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Add or remove context menu items in Angular Context menu component

ContextMenu items can be added or removed using the [insertAfter](#), [insertBefore](#) and [removeItems](#) methods.

In the following example, the **Display Settings** menu items are added before the **Personalize** item, the **Sort By** menu items are added after the **Refresh**, and the **Paste** item is removed from context menu.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, ViewChild } from '@angular/core';
import { ContextMenuComponent, MenuEventArgs, MenuItemModel } from
 '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [

    ContextMenuModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!--target element-->
    <div id="target">Right click / Touch hold to open the
ContextMenu</div>
    <!-- To Render ContextMenu. -->
    <ejs-contextmenu #contextmenu target='#target'
[items]='menuItems' (created)='onCreated()'></ejs-contextmenu>
    </div>`
})
export class AppComponent {
  @ViewChild('contextmenu')
  public contextmenu?: ContextMenuComponent;
  public menuItems: MenuItemModel[] = [
    {
      text: 'View',
      items: [
        {
          text: 'Large icons'
        },
        {
          text: 'Medium icons'
        },
        {
          text: 'Small icons'
        }
      ]
    },
    {
      text: 'Refresh'
    },
    {
      text: 'Paste'
    },
    {
      separator: true
    },
    {
      text: 'New'
    },
    {
      separator: true
    }
  ]
}

```



```

    },
    {
        text: 'Personalize'
    }
  ];
  onCreated(): void {
    (this.contextmenu as ContextMenuComponent).insertAfter([
      {text: 'Sort By'}], 'Refresh');
    (this.contextmenu as ContextMenuComponent).insertBefore([
      {text: 'Display Settings'}], 'Personalize');
    (this.contextmenu as ContextMenuComponent).removeItems(['Paste']);
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Enable or disable context menu items in Angular Context menu component

You can enable and disable the menu items using the [enableItems](#) method in ContextMenu. To enable menuitems, set the `enable` property in argument to `true` and vice-versa.

In the following example, the **Display Settings** in parent items and **Medium icons** in sub menu items are disabled.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ContextMenuModule } from '@syncfusion/ej2-angular-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
import { Component, ViewChild } from '@angular/core';
import { ContextMenuComponent, MenuEventArgs, MenuItemModel } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    ContextMenuModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!--target element-->
    <div id="target">Right click / Touch hold to open the
ContextMenu</div>
    <!-- To Render ContextMenu. -->
    <ejs-contextmenu #contextmenu target="#target"
[items]='menuItems' (created)='onCreated()'
(beforeOpen)='beforeOpen()'></ejs-contextmenu>
    </div>`
})
export class AppComponent {
  @ViewChild('contextmenu')

```

```

public contextmenu?: ContextMenuComponent;
// ContextMenu items definition
public menuItems: MenuItemModel[] = [
{
    text: 'View',
    items: [
        {
            text: 'Large icons'
        },
        {
            text: 'Medium icons'
        },
        {
            text: 'Small icons'
        }
    ]
},
{
    text: 'Sort By'
},
{
    text: 'Refresh'
},
{
    separator: true
},
{
    text: 'New'
},
{
    separator: true
},
{
    text: 'Display Settings'
},
{
    text: 'Personalize'
}
];
onCreated(): void {
    (this.contextmenu as ContextMenuComponent).enableItems(['Display
Settings'], false);
}
beforeOpen(): void {
    (this.contextmenu as ContextMenuComponent).enableItems(['Medium
icons'], false);
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

To disable sub menu items, use the [beforeOpen](#) event.

Dashboard Layout

Getting started with Angular Dashboard layout component

This section explains how to create a simple **DashboardLayout** component and its basic usage.

Prerequisites

To get started with **DashboardLayout** component, ensure the compatible versions of Angular and Typescript.

- Angular : 6+
- Typescript : 2.6+

Setting up angular project

Angular provides the easiest way to set angular CLI projects using Angular CLI tool.

Install the CLI application globally to your machine by using following command.

```
`sh
npm install -g @angular/cli
`
```

Create a new angular application

```
`sh
ng new syncfusion-angular-app
`
```

Navigate to the created project folder by using following command.

```
`sh
cd syncfusion-angular-app
`
```

Refer [Syncfusion Angular Getting Started](#) section to know more about setting up **angular-cli** project.

Adding Dependencies

The following list of dependencies are required to use the DashboardLayout component in your application.

```
`javascript
|-- @syncfusion/ej2-angular-layouts
|-- @syncfusion/ej2-angular-base
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-layouts
`
```

Installing Syncfusion DashboardLayout package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(`>=20.2.36`) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-layouts](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-layouts --save
```

```
,
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-layouts@ngcc](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-layouts@ngcc --save
```

```
,
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
```

```
@syncfusion/ej2-angular-layouts:"20.2.38-ngcc"
```

```
,
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding style sheet to the application

To render the DashboardLayout component, import the DashboardLayout and its dependent component's styles as given below in `[src/styles.css]`.

```
`css
```

```
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
```

```
@import "../node_modules/@syncfusion/ej2-angular-layouts/styles/material.css";
```

```
,
```

Note: To refer the combined component styles, use Syncfusion [CRG](#) (Custom Resource Generator) in your application.

Add DashboardLayout to the application

You can render the DashboardLayout component in the following two ways.

- Defined the panels property as the attribute in the HTML element directly.
- Using the `panels` property through tag helper.

Setting the `panels` property using HTML attributes

You can render the DashboardLayout component by adding the panels property as the attribute to the HTML element directly. Add the HTML div element with panel definition for DashboardLayout by using `<ejs-dashboardlayout>` selector in `template` section of the `app.component.ts` file.

Now, modify the `template` in `app.component.ts` file to render DashboardLayout component.

[src/app/app.component.ts]

```
`typescript
```

```
import { Component } from '@angular/core';
```

```
@Component({
```

```
  selector: 'app-root',
```

```
  styleUrls: ['default-style.css'],
```

```
  template: `
```

```
<div class="control-section">
```

```
<ejs-dashboardlayout id='defaultLayout' [columns]="5" #defaultLayout [cellSpacing]='cellSpacing'>
```

```
<div id="one" class="e-panel" data-row="0" data-col="0" data-sizeX="1" data-sizeY="1">
```

```
<span id="close" class="e-template-icon e-clear-icon"></span>
```

```
<div class="e-panel-container">
```

```
<div class="text-align">0</div>
```

```
</div>
```

```
</div>
```

```
<div id="two" class="e-panel" data-row="1" data-col="0" data-sizeX="1" data-sizeY="2">
```

```
<span id="close" class="e-template-icon e-clear-icon"></span>
```

```
<div class="e-panel-container">
```

```
<div class="text-align">1</div>
```

```
</div>
```

```
</div>
```

```
<div id="three" class="e-panel" data-row="0" data-col="1" data-sizeX="2" data-sizeY="2">
```

```
<span id="close" class="e-template-icon e-clear-icon"></span>
```

```
<div class="e-panel-container">
<div class="text-align">2</div>
</div>
</div>
<div id="four" class="e-panel" data-row="2" data-col="1" data-sizeX="1" data-sizeY="1">
<span id="close" class="e-template-icon e-clear-icon"></span>
<div class="e-panel-container">
<div class="text-align">3</div>
</div>
</div>
<div id="five" class="e-panel" data-row="2" data-col="2" data-sizeX="2" data-sizeY="1">
<span id="close" class="e-template-icon e-clear-icon"></span>
<div class="e-panel-container">
<div class="text-align">4</div>
</div>
</div>
<div id="six" class="e-panel" data-row="0" data-col="3" data-sizeX="1" data-sizeY="1">
<span id="close" class="e-template-icon e-clear-icon"></span>
<div class="e-panel-container">
<div class="text-align">5</div>
</div>
</div>
<div id="seven" class="e-panel" data-row="1" data-col="3" data-sizeX="1" data-sizeY="1">
<span id="close" class="e-template-icon e-clear-icon"></span>
<div class="e-panel-container">
<div class="text-align">6</div>
</div>
</div>
<div id="eight" class="e-panel" data-row="0" data-col="4" data-sizeX="1" data-sizeY="3">
<span id="close" class="e-template-icon e-clear-icon"></span>
<div class="e-panel-container">
<div class="text-align">7</div>
</div>
```

```

</div>
</ejs-dashboardlayout>
</div>`
})
export class AppComponent {
}
`

```

- Import DashboardLayout module into Angular application(app.module.ts) from the package `@syncfusion/ej2-angular-layouts`.

```

`javascript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// import the DashboardLayoutModule for the Dashboard Layout component
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts';
import { AppComponent } from './app.component';
@NgModule({
//declaration of ej2-angular-layouts module into NgModule
imports: [ BrowserModule, DashboardLayoutModule ],
declarations: [ AppComponent ],
bootstrap: [ AppComponent ]
})
export class AppModule { }
`

```

Run the application

Now, use the `npm start` command to run the application in the browser.

```

`html
npm start
`

```

The following example shows a basic DashboardLayout by adding the panels property directly into the HTML element.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'

```

```

import { Component, ViewEncapsulation, ViewChild } from '@angular/core';
@Component({
  imports: [ DashboardLayoutModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
      <ejs-dashboardlayout id='defaultLayout' #defaultLayout [columns]="5"
[cellSpacing]='cellSpacing'>
        <div id="one" class="e-panel" data-row="0" data-col="0" data-
sizeX="1" data-sizeY="1">
          <span id="close" class="e-template-icon e-clear-icon"></span>
          <div class="e-panel-container">
            <div class="text-align">0</div>
          </div>
        </div>
        <div id="two" class="e-panel" data-row="1" data-col="0" data-
sizeX="1" data-sizeY="2">
          <span id="close" class="e-template-icon e-clear-icon"></span>
          <div class="e-panel-container">
            <div class="text-align">1</div>
          </div>
        </div>
        <div id="three" class="e-panel" data-row="0" data-col="1" data-
sizeX="2" data-sizeY="2">
          <span id="close" class="e-template-icon e-clear-icon"></span>
          <div class="e-panel-container">
            <div class="text-align">2</div>
          </div>
        </div>
        <div id="four" class="e-panel" data-row="2" data-col="1" data-
sizeX="1" data-sizeY="1">
          <span id="close" class="e-template-icon e-clear-icon"></span>
          <div class="e-panel-container">
            <div class="text-align">3</div>
          </div>
        </div>
        <div id="five" class="e-panel" data-row="2" data-col="2" data-
sizeX="2" data-sizeY="1">
          <span id="close" class="e-template-icon e-clear-icon"></span>
          <div class="e-panel-container">
            <div class="text-align">4</div>
          </div>
        </div>
        <div id="six" class="e-panel" data-row="0" data-col="3" data-
sizeX="1" data-sizeY="1">
          <span id="close" class="e-template-icon e-clear-icon"></span>
          <div class="e-panel-container">
            <div class="text-align">5</div>
          </div>
        </div>
        <div id="seven" class="e-panel" data-row="1" data-col="3" data-
sizeX="1" data-sizeY="1">
          <span id="close" class="e-template-icon e-clear-icon"></span>
          <div class="e-panel-container">
            <div class="text-align">6</div>
          </div>
        </div>
      </div>
    `
})

```



```

        </div>
      </div>
      <div id="eight" class="e-panel" data-row="0" data-col="4" data-
sizeX="1" data-sizeY="3">
        <span id="close" class="e-template-icon e-clear-icon"></span>
        <div class="e-panel-container">
          <div class="text-align">7</div>
        </div>
      </div>
    </ejs-dashboardlayout>
  </div>`,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  public cellSpacing: number[] = [10, 10];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Setting the `panels` property through binding

You can render the DashboardLayout component by using the **panels** property through binding.

Now, modify the `template` in `app.component.ts` file to render DashboardLayout component.

[src/app/app.component.ts]

`typescript

```

import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  selector: 'app-root',
  styleUrls: ['app/default-style.css'],
  template: `
<div class="control-section">
  <ejs-dashboardlayout id='defaultLayout' #defaultLayout [cellSpacing]='cellSpacing' [panels]='panels'
[columns]="5">
</ejs-dashboardlayout>
</div>`,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  public cellSpacing: number[] = [10, 10];
}

```

```

public panels: any = [{ "sizeX": 1, "sizeY": 1, "row": 0, "col": 0, content: '<div class="content">0</div>' },
{ "sizeX": 3, "sizeY": 2, "row": 0, "col": 1, content: '<div class="content">1</div>' },
{ "sizeX": 1, "sizeY": 3, "row": 0, "col": 4, content: '<div class="content">2</div>' },
{ "sizeX": 1, "sizeY": 1, "row": 1, "col": 0, content: '<div class="content">3</div>' },
{ "sizeX": 2, "sizeY": 1, "row": 2, "col": 0, content: '<div class="content">4</div>' },
{ "sizeX": 1, "sizeY": 1, "row": 2, "col": 2, content: '<div class="content">5</div>' },
{ "sizeX": 1, "sizeY": 1, "row": 2, "col": 3, content: '<div class="content">6</div>' }
]
}
,

```

The following example shows a basic DashboardLayout by using the `panels` property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [ DashboardLayoutModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
      <ejs-dashboardlayout id='defaultLayout' #defaultLayout [columns]='5'
[cellSpacing]='cellSpacing' [panels]='panels'>
        </ejs-dashboardlayout>
    </div>`,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  public cellSpacing: number[] = [10, 10];
  public panels: any = [{ "sizeX": 1, "sizeY": 1, "row": 0, "col": 0,
content: '<div class="content">0</div>' },
{ "sizeX": 3, "sizeY": 2, "row": 0, "col": 1, content: '<div
class="content">1</div>' },
{ "sizeX": 1, "sizeY": 3, "row": 0, "col": 4, content: '<div
class="content">2</div>' },
{ "sizeX": 1, "sizeY": 1, "row": 1, "col": 0, content: '<div
class="content">3</div>' },
{ "sizeX": 2, "sizeY": 1, "row": 2, "col": 0, content: '<div
class="content">4</div>' },
{ "sizeX": 1, "sizeY": 1, "row": 2, "col": 2, content: '<div
class="content">5</div>' },
{ "sizeX": 1, "sizeY": 1, "row": 2, "col": 3, content: '<div
class="content">6</div>' }
]
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

You can refer to our [Angular Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Dashboard Layout example](#) to know how to present and manipulate data.

Setting size of cells in Angular Dashboard layout component

The entire layout dimensions are assigned based on the height and width of the parent element. Hence, a responsive or static layout can be created by assigning a percentage or static dimension values to the parent element. The layout adapts to mobile resolutions by transforming the entire layout into a stacked orientation, so that, the panels will be displayed in a vertical column.

The **Dashboard Layout** is a grid structured component which can be split into subsections of equal size known as cells. The total number of cells in each row is defined by using the `columns` property of the component. The width of each cell will be auto calculated based on the total number of cells placed in a row and the height of a cell will be same as that of its width. However, the height of these cells can also be configured to any desired size using the `cellAspectRatio` property (cellwidth/cellheight ratio) which defines the cell width to height ratio.

The number of rows within the layout has no limits and can have any number of rows based on the panels count and position. Panels which acts as data containers will be placed or positioned over these cells.

Modifying cell size

In a dashboard, the data to be held by the panel in a cell may be of different size, hence different cell dimensions may be required in different scenarios. In this case, the size of these grid cells can be modified to the required size using the `columns` and `cellAspectRatio` properties.

The following sample demonstrates how to modify a cell size using the `columns` and `cellAspectRatio` properties. In the following sample, the width of the parent element is divided into 5 equal cells based on the `columns` property value resulting the width of each cell as 100px. The height of these cells will be 50px based on the `cellAspectRatio` value 100/50 (i.e., for every 100px of width, 50px will be the height of the cell).

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [ DashboardLayoutModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
```

```

    <ejs-dashboardlayout id='defaultLayout' [columns]='5'
[cellSpacing]='cellSpacing' [panels]='panels'
[cellAspectRatio]='cellAspectRatio'>
    </ejs-dashboardlayout>
</div>`,
    encapsulation: ViewEncapsulation.None
  })
  export class AppComponent {
    public cellSpacing: number[] = [10, 10];
    public cellAspectRatio: number = 100/50;
    public panels: any = [{ "sizeX": 1, "sizeY": 1, "row": 0, "col": 0,
content:'<div class="content">0</div>' },
    { "sizeX": 3, "sizeY": 2, "row": 0, "col": 1, content:'<div
class="content">1</div>' },
    { "sizeX": 1, "sizeY": 3, "row": 0, "col": 4, content:'<div
class="content">2</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 1, "col": 0, content:'<div
class="content">3</div>' },
    { "sizeX": 2, "sizeY": 1, "row": 2, "col": 0, content:'<div
class="content">4</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 2, content:'<div
class="content">5</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 3, content:'<div
class="content">6</div>' }
    ];
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Setting cell spacing

The spacing between each panel in a row and column can be defined using the `cellSpacing` property. Adding spacing between the panels will make the layout effective and provides a clear data representation.

The following sample demonstrates the usage of the `cellSpacing` property, which helps in a neat and clear representation of a data.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [ DashboardLayoutModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">

```

```

    <ejs-dashboardlayout id='defaultLayout' [columns]='5'
    [cellSpacing]='cellSpacing' [panels]='panels'>
    </ejs-dashboardlayout>
    </div>`,
    encapsulation: ViewEncapsulation.None
  })
  export class AppComponent {
    public cellSpacing: number[] = [20, 20];
    public panels: any = [{ "sizeX": 1, "sizeY": 1, "row": 0, "col": 0,
    content:'<div class="content">0</div>' },
    { "sizeX": 3, "sizeY": 2, "row": 0, "col": 1, content:'<div
    class="content">1</div>' },
    { "sizeX": 1, "sizeY": 3, "row": 0, "col": 4, content:'<div
    class="content">2</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 1, "col": 0, content:'<div
    class="content">3</div>' },
    { "sizeX": 2, "sizeY": 1, "row": 2, "col": 0, content:'<div
    class="content">4</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 2, content:'<div
    class="content">5</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 3, content:'<div
    class="content">6</div>' }
    ];
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Graphical representation of layout

These cells combinedly forms a grid-structured layout which will be hidden initially. This grid structured layout can be made visible by enabling the `showGridLines` property, which clearly pictures the cells split-up within the layout. These gridlines will be helpful in panels sizing and placement within the layout during initial designing of a dashboard.

In the following sample, the gridlines indicate the cells split-up of the layout and the data containers placed over these cells are known as panels.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [ DashboardLayoutModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">

```

```

    <ejs-dashboardlayout id='defaultLayout' [columns]='5'
[cellSpacing]='cellSpacing' [panels]='panels'
[showGridLines]='showGridLines'>
    </ejs-dashboardlayout>
</div>`,
    encapsulation: ViewEncapsulation.None
  })
  export class AppComponent {
    public cellSpacing: number[] = [10, 10];
    public showGridLines: boolean = true;
    public panels: any = [{ "sizeX": 3, "sizeY": 2, "row": 0, "col": 1,
content:'<div class="content">1</div>' },
    { "sizeX": 1, "sizeY": 3, "row": 0, "col": 4, content:'<div
class="content">2</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 2, content:'<div
class="content">3</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 3, content:'<div
class="content">4</div>' }
    ];
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Rendering component in right-to-left direction

It is possible to render the Dashboard Layout in right-to-left direction by setting the [enableRtl](#) API to true.

The following sample demonstrates Dashboard Layout in right-to-left direction.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [ DashboardLayoutModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
      <ejs-dashboardlayout id='defaultLayout' #defaultLayout
[columns]='columns' [cellSpacing]='cellSpacing' [panels]='panels'
[enableRtl]='enableRtl'>
        </ejs-dashboardlayout>
    </div>`,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  public cellSpacing: number[] = [10, 10];
}

```

```

public columns: number = 5;
public enableRtl: boolean = true;
public panels: any = [
  { 'sizeX': 1, 'sizeY': 1, 'row': 0, 'col': 0, header: '<div>Panel
0</div>', content: '<div class="content"></div>' },
  { 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 1, header: '<div>Panel
1</div>', content: '<div class="content"></div>' },
  { 'sizeX': 1, 'sizeY': 3, 'row': 0, 'col': 4, header: '<div>Panel
2</div>', content: '<div class="content"></div>' },
  { 'sizeX': 1, 'sizeY': 1, 'row': 1, 'col': 0, header: '<div>Panel
3</div>', content: '<div class="content"></div>' },
  { 'sizeX': 2, 'sizeY': 1, 'row': 2, 'col': 0, header: '<div>Panel
4</div>', content: '<div class="content"></div>' },
  { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 2, header: '<div>Panel
5</div>', content: '<div class="content"></div>' },
  { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 3, header: '<div>Panel
6</div>', content: '<div class="content"></div>' }
];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

You can refer to our [Angular Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Dashboard Layout example](#) to know how to present and manipulate data.

Panels

Position sizing of panels in Angular Dashboard layout component

Panels are the basic building blocks of the dashboard layout component. They act as a container for the data to be visualized or presented. These panels can be positioned or resized for effective presentation of the data.

The following table represents all the available panel properties and the corresponding functionalities

PanelObject	Description
---	---
id	Specifies the ID value of the panel.
row	Specifies the row value in which the panel to be placed.
col	Specifies the column value in which the panel to be placed.
sizeX	Specifies the width of the panel in cells count.
sizeY	Specifies the height of the panel in cells count.
minSizeX	Specifies the minimum width of the panel in cells count.
minSizeY	Specifies the minimum height of the panel in cells count.

- | **maxSizeX** | Specifies the maximum width of the panel in cells count. |
- | **maxSizeY** | Specifies the maximum height of the panel in cells count. |
- | **header** | Specifies the header template of the panel. |
- | **content** | Specifies the content template of the panel. |
- | **cssClass** | Specifies the CSS class name that can be appended with each panel element. |

Positioning of panels

The panels within the layout can be easily positioned or ordered using the **row** and **col** properties of the panels. Positioning of panels will be beneficial to represent the data in any desired order.

The following sample demonstrates the positioning of panels within the dashboard layout using the row and column properties of the panels.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [ DashboardLayoutModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
      <ejs-dashboardlayout id='defaultLayout' #defaultLayout
[columns]='columns' [cellSpacing]='cellSpacing' [panels]='panels'>
      </ejs-dashboardlayout>
    </div>`,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  public cellSpacing: number[] = [20, 20];
  public columns: number = 3;
  public panels: any = [{ "row": 0, "col": 0, content:'<div
class="content">1</div>' },
  { "row": 0, "col": 1, content:'<div class="content">2</div>' },
  { "row": 0, "col": 2, content:'<div class="content">3</div>' },
  { "row": 1, "col": 0, content:'<div class="content">4</div>' },
  { "row": 1, "col": 1, content:'<div class="content">5</div>' },
  { "row": 1, "col": 2, content:'<div class="content">6</div>' }
  ]
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```


Sizing of panels

A panel's size can be varied easily by defining the `sizeX` and `sizeY` properties. The `sizeX` property defines the width and the `sizeY` property defines height of a panel in cells count. These properties are helpful in designing a dashboard, where the content of each panel may vary in size.

The following sample demonstrates the sizing of panels within the dashboard layout using the `sizeX` and `sizeY` properties of the panels.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [ DashboardLayoutModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
      <ejs-dashboardlayout id='defaultLayout' #defaultLayout
[columns]='columns' [cellSpacing]='cellSpacing' [panels]='panels'>
      </ejs-dashboardlayout>
    </div>`,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  public cellSpacing: number[] = [20, 20];
  public columns: number = 5;
  public panels: any = [{ "sizeX": 1, "sizeY": 1, "row": 0, "col": 0,
content:'<div class="content">0</div>' },
  { "sizeX": 3, "sizeY": 2, "row": 0, "col": 1, content:'<div
class="content">1</div>' },
  { "sizeX": 1, "sizeY": 3, "row": 0, "col": 4, content:'<div
class="content">2</div>' },
  { "sizeX": 1, "sizeY": 1, "row": 1, "col": 0, content:'<div
class="content">3</div>' },
  { "sizeX": 2, "sizeY": 1, "row": 2, "col": 0, content:'<div
class="content">4</div>' },
  { "sizeX": 1, "sizeY": 1, "row": 2, "col": 2, content:'<div
class="content">5</div>' },
  { "sizeX": 1, "sizeY": 1, "row": 2, "col": 3, content:'<div
class="content">6</div>' }
  ]
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

You can refer to our [Angular Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Dashboard Layout example](#) to know how to present and manipulate data.

Setting header of panels in Angular Dashboard layout component

The dashboard layout component is mostly used to represent the data used for monitoring or managing a process. These data or any HTML template can be placed as the content of a panel using the `content` property. Also, word or phrase that summarize the panel's content can be added as the header on the top of each panel using the `header` property of the panel.

The following sample demonstrates how to add content for each panel using the header and content properties of the panels.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [ DashboardLayoutModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
      <ejs-dashboardlayout id='defaultLayout' #defaultLayout
        [columns]='columns' [cellSpacing]='cellSpacing' [panels]='panels'>
      </ejs-dashboardlayout>
    </div>`,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  public cellSpacing: number[] = [10, 10];
  public columns: number = 6;
  public panels: any = [{ 'id': 'Panel0', 'sizeX': 1, 'sizeY': 1, 'row': 0,
    'col': 0, header: '<div>Panel 0</div>', content: '<div class="content">Panel
    Content<div>',
    { 'id': 'Panel1', 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 1,
    header: '<div>Panel 1</div>', content: '<div class="content">Panel
    Content<div>',
    { 'id': 'Panel2', 'sizeX': 1, 'sizeY': 3, 'row': 0, 'col': 4,
    header: '<div>Panel 2</div>', content: '<div class="content">Panel
    Content<div>',
    { 'id': 'Panel3', 'sizeX': 1, 'sizeY': 1, 'row': 1, 'col': 0,
    header: '<div>Panel 3</div>', content: '<div class="content">Panel
    Content<div>',
    { 'id': 'Panel4', 'sizeX': 2, 'sizeY': 1, 'row': 2, 'col': 0,
    header: '<div>Panel 4</div>', content: '<div class="content">Panel
    Content<div>',
    { 'id': 'Panel5', 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 2,
    header: '<div>Panel 5</div>', content: '<div class="content">Panel
    Content<div>',
    { 'id': 'Panel6', 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 3,
    header: '<div>Panel 6</div>', content: '<div class="content">Panel
    Content<div>' }
```

```
]
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Placing components as content of panels

In a dashboard, components like the chart, grids, maps, gauge and more . can be used to present a complex data. Such components can be placed as the panel content by assigning the corresponding component element as the **content** of the panel.

The following sample demonstrates how to add ej2-chart components as the **content** for each panel in the dashboard layout component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'
import { ChartAllModule, AccumulationChartAllModule } from '@syncfusion/ej2-angular-charts'
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [ DashboardLayoutModule, ChartAllModule,
    AccumulationChartAllModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
      <ejs-dashboardlayout [columns]="6" #editLayout
[cellSpacing]='cellSpacing'>
        <e-panels>
          <e-panel [sizeX]="3" [sizeY]="2" [row]="0" [col]="0">
            <ng-template #header>
              <div>Product usage ratio</div>
            </ng-template>
            <ng-template #content>
              <div id="column">
                <ejs-chart id="columnChart" height="162px"
[primaryXAxis]='primaryXAxis'>
                  <e-series-collection>
                    <e-series [dataSource]="chartData"
type='Column' xName='month' yName='sales'>
                  </e-series>
                </e-series-collection>
              </ejs-chart>
            </div>
          </ng-template>
        </e-panel>
        <e-panel [sizeX]="3" [sizeY]="2" [row]="0" [col]="3">
```

```

        <ng-template #header>
            <div>Last year Sales Comparison</div>
        </ng-template>
        <ng-template #content>
            <div id="line">
                <ejs-chart id="lineChart" height="162px"
[primaryXAxis]='primaryXAxis'>
                    <e-series-collection>
                        <e-series [dataSource]="lineData"
xName='x' yName='y' type="Line">
                            </e-series>
                        </e-series-collection>
                    </ejs-chart>
                </div>
            </ng-template>
        </e-panel>
        <e-panel [sizeX]="3" [sizeY]="2" [row]="0" [col]="3">
            <ng-template #header>
                <div>Sales increase percentage 1</div>
            </ng-template>
            <ng-template #content>
                <div id="pie">
                    <ejs-accumulationchart id="pieChart"
height="162px" [legendSettings]="legendSettings" [tooltip]='tooltip'>
                        <e-accumulation-series-collection>
                            <e-accumulation-series
[dataSource]="piechart" xName="x" yName="y" innerRadius="20%"
                                name="Browser"
[dataLabel]='datalabel'>
                                    </e-accumulation-series>
                                </e-accumulation-series-collection>
                            </ejs-accumulationchart>
                        </div>
                    </ng-template>
                </e-panel>
                <e-panel [sizeX]="3" [sizeY]="2" [row]="1" [col]="0">
                    <ng-template #header>
                        <div>Sales increase percentage 2</div>
                    </ng-template>
                    <ng-template #content>
                        <div id="pie1">
                            <ejs-accumulationchart id="pieChart1"
[enableAnimation]="false" height="162px"
                                [tooltip]='tooltip'
[legendSettings]='legendSettings'>
                                    <e-accumulation-series-collection>
                                        <e-accumulation-series
[dataSource]="piechart1" xName="x" yName="y" radius="70%"
                                            name="Browser"
[dataLabel]='datalabel'>
                                                </e-accumulation-series>
                                            </e-accumulation-series-collection>
                                        </ejs-accumulationchart>
                                    </div>
                                </ng-template>
                            </e-panel>
                        </e-panels>

```

```

        </ejs-dashboardlayout>
    </div>`,
    encapsulation: ViewEncapsulation.None
})
export class AppComponent {
    public cellSpacing: number[] = [10, 10];
    public chartData: Object[] = [
        { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
        { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
        { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
        { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
        { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
        { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
    ];
    public primaryXAxis: Object = {
        valueType: 'Category'
    }
    public lineData: any[] = [
        { x: 2013, y: 28 }, { x: 2014, y: 25 }, { x: 2015, y: 26 }, { x: 2016, y:
27 },
        { x: 2017, y: 32 }, { x: 2018, y: 35 }
    ];
    public piechart: any[] = [{ x: 'TypeScript', y: 13, text: 'TS 13%' }, {
x: 'React', y: 12.5, text: 'React 12.5%' }, { x: 'MVC', y: 12, text: 'MVC 12%'
}, { x: 'Core', y: 12.5, text: 'Core 12.5%' }, { x: 'Vue', y: 10, text: 'Vue
10%' }, { x: 'Angular', y: 40, text: 'Angular 40%' }];
    public piechart1: any[] = [
        { 'x': 'Chrome', y: 37, text: '37%' },
        { 'x': 'UC Browser', y: 17, text: '17%' },
        { 'x': 'iPhone', y: 19, text: '19%' },
        { 'x': 'Others', y: 4, text: '4%' },
        { 'x': 'Opera', y: 11, text: '11%' },
        { 'x': 'Android', y: 12, text: '12%' }
    ];
    public legendSettings: Object = {
        visible: false
    };
    tooltip: any;
    datalabel: any;
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

You can refer to our [Angular Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Dashboard Layout example](#) to know how to present and manipulate data.

Add remove panels in Angular Dashboard layout component

In real-time cases, the data being presented within the dashboard should be updated frequently which includes adding or removing the data dynamically within the dashboard. This can be easily achieved by using the `addPanel` and `removePanel` public methods of the component.

Add or remove panels dynamically

Panels can be added dynamically by using the `addPanel` public method by passing the `panel` property as parameter. Also, they can be removed dynamically by using the `removePanel` public method by passing the `panel id` value as a parameter.

It is also possible to remove all the panels in a Dashboard Layout by calling `removeAll` method.

```
`js
```

```
dashboard.removeAll();
```

The following sample demonstrates how to add and remove the panels dynamically in the dashboard layout component. Here, panels can be added in any desired position of required size by selecting them in the numeric boxes and clicking add button and remove them by selecting the ID of the panel.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { NumericTextBoxModule } from '@syncfusion/ej2-angular-inputs'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, ViewEncapsulation, ViewChild } from '@angular/core';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
import { NumericTextBoxComponent } from '@syncfusion/ej2-angular-inputs';
import { DropDownList, DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns';
import { DashboardLayoutComponent } from '@syncfusion/ej2-angular-layouts';
@Component({
  imports: [ DashboardLayoutModule, ButtonModule, NumericTextBoxModule,
    DropDownListModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
      <div class="inline" id="control">
        <ejs-dashboardlayout id='defaultLayout' #defaultLayout
        [columns]='columns' [cellSpacing]='cellSpacing'
          [panels]='panels'>
        </ejs-dashboardlayout>
      </div>
      <div class="inline" id="properties">
        <table>
          <tr>
            <td>SizeX</td>
            <td>
```

```

        <ejs-numerictextbox id="sizeX" #sizeX
placeholder="Ex: 1" floatLabelType="Never" value= 1 min=1 max=5></ejs-
numerictextbox>
    </td>
</tr>
<tr>
    <td>SizeY</td>
    <td>
        <ejs-numerictextbox id="sizeY" #sizeY
placeholder="Ex: 1" floatLabelType="Never" value= 1 min=1 max=5></ejs-
numerictextbox>
    </td>
</tr>
<tr>
    <td>Row</td>
    <td>
        <ejs-numerictextbox id="row" #row placeholder="Ex: 1"
floatLabelType="Never" value=0 min=0 max=5></ejs-numerictextbox>
    </td>
</tr>
<tr>
    <td>Column</td>
    <td>
        <ejs-numerictextbox id="column" #column
placeholder="Ex: 1" floatLabelType="Never" value=0 min=0 max=4></ejs-
numerictextbox>
    </td>
</tr>
<tr>
    <td> </td>
    <td>
        <button ej-button id="add" #add cssClass="e-btn e-
flat" (click)="addClick()">Add</button>
    </td>
</tr>
</table>
<table>
    <tr>
        <td>Id</td>
        <td> <ejs-dropdownlist id='dropdown' #dropdown
[dataSource]='data' placeholder='Select a id value'
></ejs-dropdownlist></td>
    </tr>
    <tr>
        <td> </td>
        <td>
            <button ej-button id="remove" cssClass="e-btn e-flat
e-danger" (click)="removeClick()">Remove</button>
        </td>
    </tr>
</table>

</div>
</div>`,
    encapsulation: ViewEncapsulation.None
})
export class AppComponent {

```

```

    @ViewChild('defaultLayout') dashboard?: DashboardLayoutComponent;
    @ViewChild('sizeX') sizeX?: NumericTextBoxComponent;
    @ViewChild('sizeY') sizeY?: NumericTextBoxComponent;
    @ViewChild('row') row?: NumericTextBoxComponent;
    @ViewChild('column') column?: NumericTextBoxComponent;
    @ViewChild('add') addBtn?: ButtonComponent;
    @ViewChild('dropdown') dropDownListObject?: DropDownListComponent;
    public data: string[] = ['Panel0', 'Panel1', 'Panel2', 'Panel3',
'Panel4', 'Panel5', 'Panel6'];
    public cellSpacing: number[] = [20, 20];
    public columns: number = 5;
    public count: number = 7;
    public panel?: any;
    public panels: any = [{ 'id': 'Panel0', 'sizeX': 1, 'sizeY': 1, 'row': 0,
'col': 0, content: '<div class="content">0</div>' },
    { 'id': 'Panel1', 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 1, content: '<div
class="content">1</div>' },
    { 'id': 'Panel2', 'sizeX': 1, 'sizeY': 3, 'row': 0, 'col': 4, content: '<div
class="content">2</div>' },
    { 'id': 'Panel3', 'sizeX': 1, 'sizeY': 1, 'row': 1, 'col': 0, content: '<div
class="content">3</div>' },
    { 'id': 'Panel4', 'sizeX': 2, 'sizeY': 1, 'row': 2, 'col': 0, content: '<div
class="content">4</div>' },
    { 'id': 'Panel5', 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 2, content: '<div
class="content">5</div>' },
    { 'id': 'Panel6', 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 3, content: '<div
class="content">6</div>' }
    ];
    addClick() {
        this.panel = {
            id: "Panel" + this.count.toString(),
            sizeX: this.sizeX?.value,
            sizeY: this.sizeY?.value,
            row: this.row?.value,
            col: this.column?.value,
            content: "<div class='content'>" + this.count + "</div>"
        }
        this.dashboard?.addPanel(this.panel);
        this.count = this.count + 1;
        (<any>this.dropDownListObject?.dataSource).push(this.panel.id);
        this.dropDownListObject?.refresh();
    };
    removeClick() {
        this.dashboard?.removePanel(<any>this.dropDownListObject?.value);

        (<any>this.dropDownListObject?.dataSource).splice((<any>this.dropDownListObje
ct?.dataSource).indexOf(this.dropDownListObject?.value), 1);
        this.dropDownListObject?.refresh();
        (this.dropDownListObject as DropDownListComponent).value = null as
any;
    };
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
```



```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

You can refer to our [Angular Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Dashboard Layout example](#) to know how to present and manipulate data.

Interaction With Panels

Dragging moving of panels in Angular Dashboard layout component

The Dashboard Layout component is provided with dragging functionality to drag and reorder the panels within the layout. While dragging a panel, a holder will be highlighted below the panel indicating the panel placement on panel drop. This helps the user to decide whether to place the panel in the current position or revert to previous position without disturbing the layout.

If one or more panels collide while dragging, then the colliding panels will be pushed towards the left or right or top or bottom direction where an adaptive space for the collided panel is available. The position changes of these collided panels will be updated dynamically during dragging of a panel, so the user can conclude whether to place the panel in the current position or not.

While dragging a panel in Dashboard layout the following dragging events will be triggered,

- [dragStart](#) - Triggers when panel drag starts
- [drag](#) - Triggers when panel is being dragged
- [dragStop](#) - Triggers when panel drag stops

The following sample demonstrates dragging and pushing of panels. For example, while dragging the panel 0 over panel 1, these panels get collided and push the panel 1 towards the feasible direction, so that, the panel 0 gets placed in the panel 1 position.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [ DashboardLayoutModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
      <ejs-dashboardlayout id='defaultLayout' #defaultLayout
        [columns]='columns' [cellSpacing]='cellSpacing' [panels]='panels'
        (dragStart)="onDragStart($this)" (drag)="onDrag($this)"
        (dragStop)="onDragStop($this)">
      </ejs-dashboardlayout>
    </div>`,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  public cellSpacing: number[] = [10, 10];
```

```

    public columns: number = 5;
    public panels: any = [{ 'sizeX': 1, 'sizeY': 1, 'row': 0, 'col': 0,
content: '<div class="content">0</div>' },
    { 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 1, content: '<div
class="content">1</div>' },
    { 'sizeX': 1, 'sizeY': 3, 'row': 0, 'col': 4, content: '<div
class="content">2</div>' },
    { 'sizeX': 1, 'sizeY': 1, 'row': 1, 'col': 0, content: '<div
class="content">3</div>' },
    { 'sizeX': 2, 'sizeY': 1, 'row': 2, 'col': 0, content: '<div
class="content">4</div>' },
    { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 2, content: '<div
class="content">5</div>' },
    { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 3, content: '<div
class="content">6</div>' },
    ];
    $this: any = this;
    //Dashboard Layout's drag start event function
    onDragStart(args: any) {
        console.log("Drag start");
    }
    //Dashboard Layout's drag event function
    onDrag(args: any) {
        console.log("Dragging");
    }
    //Dashboard Layout's dragstop event function
    onDragStop(args: any) {
        console.log("Drag stop");
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customizing the dragging handler

Initially, the complete panel will act as the handler for dragging the panel such that the dragging action occurs on clicking anywhere over a panel. However, this dragging handler for the panels can be customized using the `draggableHandle` property to restrict the dragging action within a particular element in the panel.

The following sample demonstrates customizing the dragging handler of the panels where the dragging action of panel occurs only with the header of the panel.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'
import { ChartAllModule, AccumulationChartAllModule } from '@syncfusion/ej2-
angular-charts'
import { Component, ViewEncapsulation } from '@angular/core';

```

```

@Component ({
  imports: [ DashboardLayoutModule, ChartAllModule,
    AccumulationChartAllModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
      <ejs-dashboardlayout [columns]="6" #editLayout
[cellSpacing]='cellSpacing' [draggableHandle]='draggableHandle'>
        <e-panels>
          <e-panel [sizeX]="3" [sizeY]="2" [row]="0" [col]="0">
            <ng-template #header>
              <div class="header">Product usage ratio</div>
              <span class="handler e-icons burg-icon"></span>
            </ng-template>
            <ng-template #content>
              <div id="column">
                <ejs-chart id="columnChart" height="162px"
[primaryXAxis]='primaryXAxis'>
                  <e-series-collection>
                    <e-series [dataSource]="chartData"
xName='month' yName='sales' type="Column">
                      </e-series>
                    </e-series-collection>
                  </ejs-chart>
                </div>
              </ng-template>
            </e-panel>
            <e-panel [sizeX]="3" [sizeY]="2" [row]="0" [col]="3">
              <ng-template #header>
                <div class="header">Last year Sales
Comparison</div>
                <span class="handler e-icons burg-icon"></span>
              </ng-template>
              <ng-template #content>
                <div id="line">
                  <ejs-chart id="lineChart" height="162px"
[primaryXAxis]='primaryXAxis'>
                    <e-series-collection>
                      <e-series [dataSource]="lineData"
xName='x' yName='y' type="Line">
                        </e-series>
                    </e-series-collection>
                  </ejs-chart>
                </div>
              </ng-template>
            </e-panel>
            <e-panel [sizeX]="3" [sizeY]="2" [row]="0" [col]="3">
              <ng-template #header>
                <div class="header">Sales increase percentage
1</div>
                <span class="handler e-icons burg-icon"></span>
              </ng-template>
              <ng-template #content>
                <div id="pie">

```

```

        <ejs-accumulationchart id="pieChart"
height="162px" [legendSettings]="legendSettings" [tooltip]='tooltip'>
        <e-accumulation-series-collection>
        <e-accumulation-series
[dataSource]="piechart" xName="x" yName="y" innerRadius="20%"
        name="Browser">
        </e-accumulation-series>
        </e-accumulation-series-collection>
        </ejs-accumulationchart>
    </div>
</ng-template>
</e-panel>
<e-panel [sizeX]="3" [sizeY]="2" [row]="1" [col]="0">
    <ng-template #header>
        <div class="header">Sales increase percentage
2</div>

        <span class="handler e-icons burg-icon"></span>
    </ng-template>
    <ng-template #content>
        <div id="pie1">
            <ejs-accumulationchart id="pieChart1"
[enableAnimation]="false" height="162px"
            [tooltip]='tooltip'
[legendSettings]='legendSettings'>
            <e-accumulation-series-collection>
            <e-accumulation-series
[dataSource]="piechart1" xName="x" yName="y" radius="70%"
            name="Browser">
            </e-accumulation-series>
            </e-accumulation-series-collection>
            </ejs-accumulationchart>
        </div>
    </ng-template>
</e-panel>
</e-panels>
</ejs-dashboardlayout>
</div>`,
    encapsulation: ViewEncapsulation.None
})
export class AppComponent {
    public cellSpacing: number[] = [10, 10];
    public draggableHandle: string = '.e-panel-header';
    public primaryXAxis: Object = { valueType: 'Category' };
    public chartData: any[] = [
        { month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },
        { month: 'Mar', sales: 34 }, { month: 'Apr', sales: 32 },
        { month: 'May', sales: 40 }, { month: 'Jun', sales: 32 },
        { month: 'Jul', sales: 35 }, { month: 'Aug', sales: 55 },
        { month: 'Sep', sales: 38 }, { month: 'Oct', sales: 30 },
        { month: 'Nov', sales: 25 }, { month: 'Dec', sales: 32 }
    ];
    public lineData: any[] = [
        { x: 2013, y: 28 }, { x: 2014, y: 25 }, { x: 2015, y: 26 }, { x:
2016, y: 27 },
        { x: 2017, y: 32 }, { x: 2018, y: 35 },
    ];
    public piechart: any[] = [

```

```

    { x: 'TypeScript', y: 13, text: 'TS 13%' },
    { x: 'React', y: 12.5, text: 'React 12.5%' },
    { x: 'MVC', y: 12, text: 'MVC 12%' },
    { x: 'Core', y: 12.5, text: 'Core 12.5%' },
    { x: 'Vue', y: 10, text: 'Vue 10%' },
    { x: 'Angular', y: 40, text: 'Angular 40%' }
  ];
  public piechart1: any[] = [
    { 'x': 'Chrome', y: 37, text: '37%' },
    { 'x': 'UC Browser', y: 17, text: '17%' },
    { 'x': 'iPhone', y: 19, text: '19%' },
    { 'x': 'Others', y: 4, text: '4%' },
    { 'x': 'Opera', y: 11, text: '11%' },
    { 'x': 'Android', y: 12, text: '12%' }
  ];
  public legendSettings: Object = {
    visible: false
  };
  tooltip: any;
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Disable dragging of panels

By default, the dragging of panels is enabled in Dashboard Layout. It can also be disabled with the help of [allowDragging](#) API. Setting [allowDragging](#) to false disables the dragging functionality in Dashboard Layout.

The following sample demonstrates Dashboard Layout with dragging support disabled.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts';
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [ DashboardLayoutModule ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
      <ejs-dashboardlayout id='defaultLayout' #defaultLayout
        [columns]='columns' [cellSpacing]='cellSpacing' [panels]='panels'
        [allowDragging]='allowDragging'>
      </ejs-dashboardlayout>
    </div>`,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {

```

```

public cellSpacing: number[] = [10, 10];
public columns: number = 5;
public allowDragging: boolean = false;
public panels: any = [
  { 'sizeX': 1, 'sizeY': 1, 'row': 0, 'col': 0, content: '<div
class="content">0</div>' },
  { 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 1, content: '<div
class="content">1</div>' },
  { 'sizeX': 1, 'sizeY': 3, 'row': 0, 'col': 4, content: '<div
class="content">2</div>' },
  { 'sizeX': 1, 'sizeY': 1, 'row': 1, 'col': 0, content: '<div
class="content">3</div>' },
  { 'sizeX': 2, 'sizeY': 1, 'row': 2, 'col': 0, content: '<div
class="content">4</div>' },
  { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 2, content: '<div
class="content">5</div>' },
  { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 3, content: '<div
class="content">6</div>' },
];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

You can refer to our [Angular Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Dashboard Layout example](#) to know how to present and manipulate data.

Moving panels in Angular Dashboard layout component

Other than drag and drop, it is possible to move the panels in Dashboard Layout programmatically. This can be achieved using [movePanel](#) method. The method is invoked as follows,

```
`js
```

```
movePanel(id, row, col)
```

```
,
```

Where,

- id - ID of the panel which needs to be moved.
- row - New row position for moving the panel.
- col - New column position for moving the panel.

Each time a panel's position is changed (Programmatically or through UI interaction), the Dashboard Layout's [change](#) event will be triggered.

The following sample demonstrates moving a panel programmatically to a new position in the Dashboard Layout's [created](#) event.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewEncapsulation, ViewChild } from '@angular/core';
import { DashboardLayoutComponent } from '@syncfusion/ej2-angular-layouts';
@Component({
  imports: [ DashboardLayoutModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
      <ejs-dashboardlayout id='defaultLayout' #defaultLayout
[columns]='columns' [cellSpacing]='cellSpacing' [panels]='panels'
      (created)="onCreated($this)" (change)="onChange($this)">
    </ejs-dashboardlayout>
    </div>`,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  @ViewChild('defaultLayout') dashboardLayout?: DashboardLayoutComponent;
  public cellSpacing: number[] = [10, 10];
  public columns: number = 5;
  public panels: any = [
    { 'sizeX': 1, 'sizeY': 1, 'row': 0, 'col': 0, content: '<div
class="content">0</div>' },
    { 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 1, content: '<div
class="content">1</div>' },
    { 'sizeX': 1, 'sizeY': 3, 'row': 0, 'col': 4, content: '<div
class="content">2</div>' },
    { 'sizeX': 1, 'sizeY': 1, 'row': 1, 'col': 0, content: '<div
class="content">3</div>' },
    { 'sizeX': 2, 'sizeY': 1, 'row': 2, 'col': 0, content: '<div
class="content">4</div>' },
    { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 2, content: '<div
class="content">5</div>' },
    { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 3, content: '<div
class="content">6</div>' },
  ];
  $this: any;
  //Dashboard Layout's created event function
  onCreated(args: any) {
    // movePanel("id", row, col)
    this.dashboardLayout?.movePanel("layout_0", 1, 0);
  }
  //Dashboard Layout's change event function
  onChange(args: any) {
    console.log("Change event triggered");
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';

```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

You can refer to our [Angular Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Dashboard Layout example](#) to know how to present and manipulate data.

Resizing of panels in Angular Dashboard layout component

The DashboardLayout component is also provided with the resizing functionality that can be enabled using the `allowResize` property of the component. This functionality allows you to resize the panels dynamically using the resizing handlers which controls the resizing of panels in various directions.

Initially, the panels can be resized only in south-east direction. However, panels can also be resized in east, west, north, south and south-west directions also by defining the required directions with the `resizableHandles` property.

On resizing a panel in Dashboard layout the following events will be triggered,

- [resizeStart](#) - Triggers when panel resize starts
- [resize](#) - Triggers when panel is being resized
- [resizeStop](#) - Triggers when panel resize stops

The following sample demonstrates how to enable and disable the resizing of panels in the DashboardLayout component in different directions.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [ DashboardLayoutModule ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
      <ejs-dashboardlayout id='defaultLayout' #defaultLayout [columns]='5'
[cellSpacing]='cellSpacing' [panels]='panels' [allowResizing]='allowResizing'
[resizableHandles]='resizableHandles'
      (resizeStart)="onResizeStart($this)" (resize)="onResize($this)"
(resizeStop)="onResizeStop($this)">
      </ejs-dashboardlayout>
    </div>`,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  public cellSpacing: number[] = [10, 10];
  public columns: number = 5;
  public allowResizing: boolean = true;
  public resizableHandles: string[] = ['e-south-east', 'e-east', 'e-west', 'e-north', 'e-south'];
  public panels: any = [{ "sizeX": 1, "sizeY": 1, "row": 0, "col": 0,
content: '<div class="content">0</div>' },
```



```

    { "sizeX": 3, "sizeY": 2, "row": 0, "col": 1, content: '<div
class="content">1</div>' },
    { "sizeX": 1, "sizeY": 3, "row": 0, "col": 4, content: '<div
class="content">2</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 1, "col": 0, content: '<div
class="content">3</div>' },
    { "sizeX": 2, "sizeY": 1, "row": 2, "col": 0, content: '<div
class="content">4</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 2, content: '<div
class="content">5</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 3, content: '<div
class="content">6</div>' }
  ];
  public $this: any = this;
  //Dashboard Layout's resizestart event function
  onResizeStart(args: any) {
    console.log("Resize start");
  }
  //Dashboard Layout's resize event function
  onResize(args: any) {
    console.log("Resizing");
  }
  //Dashboard Layout's resizestop event function
  onResizeStop(args: any) {
    console.log("Resize stop");
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Resizing panels programmatically

The Dashboard Layout panels can also be resized programmatically by using [resizePanel](#) method. The method is invoked as follows,

```

`js
resizePanel(id, sizeX, sizeY)
`

```

Where,

- id - ID of the panel which needs to be resized.
- sizeX - New panel width in cells count for resizing the panel.
- sizeY - New panel height in cells count for resizing the panel.

The following sample demonstrates resizing panels programmatically in the Dashboard Layout's [created](#) event.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewEncapsulation, ViewChild } from '@angular/core';
import { DashboardLayoutComponent } from '@syncfusion/ej2-angular-layouts';
@Component({
  imports: [ DashboardLayoutModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
      <ejs-dashboardlayout id='defaultLayout' #defaultLayout
[columns]='columns' [allowResizing]='true' [cellSpacing]='cellSpacing'
[panels]='panels'
      (created)="onCreated($this)">
    </ejs-dashboardlayout>
    </div>`,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  @ViewChild('defaultLayout') dashboardLayout?: DashboardLayoutComponent;
  public cellSpacing: number[] = [10, 10];
  public columns: number = 5;
  public panels: any = [
    { 'sizeX': 1, 'sizeY': 1, 'row': 0, 'col': 0, content: '<div
class="content">0</div>' },
    { 'sizeX': 3, 'sizeY': 2, 'row': 0, 'col': 1, content: '<div
class="content">1</div>' },
    { 'sizeX': 1, 'sizeY': 3, 'row': 0, 'col': 4, content: '<div
class="content">2</div>' },
    { 'sizeX': 1, 'sizeY': 1, 'row': 1, 'col': 0, content: '<div
class="content">3</div>' },
    { 'sizeX': 2, 'sizeY': 1, 'row': 2, 'col': 0, content: '<div
class="content">4</div>' },
    { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 2, content: '<div
class="content">5</div>' },
    { 'sizeX': 1, 'sizeY': 1, 'row': 2, 'col': 3, content: '<div
class="content">6</div>' },
  ];
  public $this: any = this;
  //Dashboard Layout's created event function
  onCreated(args: any) {
    // resizePanel("id", sizeX, sizeY)
    this.dashboardLayout?.resizePanel("layout_4", 1, 1);
    this.dashboardLayout?.resizePanel("layout_5", 2, 1);
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

You can refer to our [Angular Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Dashboard Layout example](#) to know how to present and manipulate data.

Floating of panels in Angular Dashboard layout component

The floating functionality of the component allows you to effectively use the entire layout for the panel's placement. If the floating functionality is enabled, the panels within the layout get floated upwards automatically to occupy the empty cells available in previous rows. This functionality can be enabled or disabled using the `allowFloating` property of the component.

The following sample demonstrates how to enable or disable the floating of panels in the DashboardLayout component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewChild, ViewEncapsulation } from '@angular/core';
import { DashboardLayoutComponent } from '@syncfusion/ej2-angular-layouts';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [ DashboardLayoutModule, ButtonModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
      <div class="inline" id="control">
        <ejs-dashboardlayout id='dashboard_default' #defaultLayout
        [columns]='6' [cellSpacing]='cellSpacing'
          [panels]='panels' [allowFloating]='allowFloating'
        [cellAspectRatio]='cellAspectRatio'>
        </ejs-dashboardlayout>
      </div>
      <div class="inline" id="properties">
        <button ej2-button id='toggle' #toggleBtn cssClass="e-flat e-
        primary e-outline" [isToggle]="true" (click)="btnClick($event)"
        content="Enable Floating"></button>
      </div>
    </div>`,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  @ViewChild('defaultLayout') dashboard?: DashboardLayoutComponent;
  @ViewChild('toggleBtn') toggleBtn?: ButtonComponent;
  public cellSpacing: any = [10, 10];
  public allowFloating: boolean = false;
  public cellAspectRatio: number = 100/75;
  public panels: any = [
    { 'sizeX': 2, 'sizeY': 2, 'row': 1, 'col': 0, content: '<div
    class="content">0</div>' },
    { 'sizeX': 2, 'sizeY': 2, 'row': 2, 'col': 2, content: '<div
    class="content">1</div>' },
```

```

    {'sizeX': 2, 'sizeY': 2, 'row': 3, 'col': 4, content: '<div
class="content">2</div>'}
  ];
  btnClick(args: any) {
    if (this.toggleBtn?.content == "Disable Floating and Reset") {
      this.toggleBtn!.content = 'Enable Floating';
      this.dashboard!.allowFloating = false;
      this.dashboard!.panels = this.panels;
    } else {
      this.toggleBtn!.content = 'Disable Floating and Reset';
      this.dashboard!.allowFloating = true;
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

You can refer to our [Angular Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Dashboard Layout example](#) to know how to present and manipulate data.

Responsive adaptive in Angular Dashboard layout component

The control is provided with built-in responsive support, where panels within the layout get adjusted based on their parent element's dimensions to accommodate any resolution which relieves the burden of building responsive dashboards.

The dashboard layout is designed to automatically adapt with lower resolutions by transforming the entire layout into a stacked one, so that, the panels will be displayed in a vertical column. By default, whenever the screen resolution meets 600px or lower resolutions this layout transformation occurs. This transformation can be modified for any user defined resolution by defining the `mediaQuery` property of the component.

The following sample demonstrates the usage of the `mediaQuery` property to turn out the layout into a stacked one in user defined resolution. Here, whenever, the window size reaches 700px or lesser, the layout becomes a stacked layout.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts';
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [ DashboardLayoutModule ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">

```

```

    <ejs-dashboardlayout id='dashboard_default' [columns]='5'
    [cellSpacing]='cellSpacing' [panels]='panels' [mediaQuery]='mediaQuery'>
    </ejs-dashboardlayout>
    </div>`,
    encapsulation: ViewEncapsulation.None
  })
  export class AppComponent {
    public cellSpacing: number[] = [10, 10];
    public mediaQuery: string = 'max-width: 700px';
    public panels: any = [{ "sizeX": 1, "sizeY": 1, "row": 0, "col": 0,
    content: '<div class="content">0</div>' },
    { "sizeX": 3, "sizeY": 2, "row": 0, "col": 1, content: '<div
    class="content">1</div>' },
    { "sizeX": 1, "sizeY": 3, "row": 0, "col": 4, content: '<div
    class="content">2</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 1, "col": 0, content: '<div
    class="content">3</div>' },
    { "sizeX": 2, "sizeY": 1, "row": 2, "col": 0, content: '<div
    class="content">4</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 2, content: '<div
    class="content">5</div>' },
    { "sizeX": 1, "sizeY": 1, "row": 2, "col": 3, content: '<div
    class="content">6</div>' }
    ]
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

You can refer to our [Angular Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Dashboard Layout example](#) to know how to present and manipulate data.

Save restore in Angular Dashboard layout component

The current layout structure of the Dashboard Layout component can be obtained and saved to construct another dashboard with same panel structure using the `serialize` public method of the component. This method returns the component's current panel setting which can be used to construct a dashboard with the same layout settings.

The following sample demonstrates how to save and restore the state of the panels using the `serialize` method. Click `Save` to store the panel's settings and click `Restore` to restore the previously saved panel settings.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { Component, ViewEncapsulation, ViewChild } from '@angular/core';

```

```

import { DashboardLayoutComponent, PanelModel } from '@syncfusion/ej2-angular-layouts';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [ DashboardLayoutModule, ButtonModule ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
      <div class="inline" id="control">
        <ejs-dashboardlayout id='dashboard_default' #defaultLayout
[columns]='5' [cellSpacing]='cellSpacing'
[panels]='panels'>
          </ejs-dashboardlayout>
        </div>
        <div class="inline" id="properties">
          <button ej-button id='saveBtn' #saveBtn cssClass="e-primary"
(click)='onSaveClick($event)'>Save</button>
          <button ej-button id='restoreBtn' #restoreBtn cssClass="e-flat
e-outline" (click)='onrestoreClick($event)'>Restore</button>
        </div>
      </div>`,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  @ViewChild('defaultLayout') dashboard?: DashboardLayoutComponent;
  @ViewChild('saveBtn') saveBtn?: ButtonComponent;
  @ViewChild('restoreBtn') restoreBtn?: ButtonComponent;
  public restoreModel: any = [];
  public cellSpacing: number[] = [10, 10];
  public panels: any = [{ "sizeX": 1, "sizeY": 1, "row": 0, "col": 0,
content: '<div class="content">0</div>' },
  { "sizeX": 3, "sizeY": 2, "row": 0, "col": 1, content: '<div
class="content">1</div>' },
  { "sizeX": 1, "sizeY": 3, "row": 0, "col": 4, content: '<div
class="content">2</div>' },
  { "sizeX": 1, "sizeY": 1, "row": 1, "col": 0, content: '<div
class="content">3</div>' },
  { "sizeX": 2, "sizeY": 1, "row": 2, "col": 0, content: '<div
class="content">4</div>' },
  { "sizeX": 1, "sizeY": 1, "row": 2, "col": 2, content: '<div
class="content">5</div>' },
  { "sizeX": 1, "sizeY": 1, "row": 2, "col": 3, content: '<div
class="content">6</div>' }
  ];
  onSaveClick(args: any) {
    this.restoreModel= this.dashboard?.serialize();
    this.restoreModel[0].content = '<div class="content">0</div>';
    this.restoreModel[1].content = '<div class="content">1</div>';
    this.restoreModel[2].content = '<div class="content">2</div>';
    this.restoreModel[3].content = '<div class="content">3</div>';
    this.restoreModel[4].content = '<div class="content">4</div>';
    this.restoreModel[5].content = '<div class="content">5</div>';
    this.restoreModel[6].content = '<div class="content">6</div>';
  }
  onrestoreClick(args: any) {

```

```

        this.dashboard!.panels = this.restoreModel;
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

You can refer to our [Angular Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Dashboard Layout example](#) to know how to present and manipulate data.

Style in Angular Dashboard layout component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the dashboard layout panel header

Use the following CSS to customize the dashboard layout panel header.

```

`css
.e-dashboardlayout.e-control .e-panel .e-panel-container .e-panel-header {
color: #754131;
background-color: #c9e2f7;
text-align: center;
}
`

```

Customizing the dashboard layout panel content

Use the following CSS to customize the dashboard layout panel content.

```

`css
.e-dashboardlayout.e-control .e-panel .e-panel-container .e-panel-content {
background-color: #c9e2f7;
padding: 50px;
}
`

```

Customizing the dashboard layout panel resize icon

Use the following CSS to customize the dashboard layout resize icon.

```

`css
.e-dashboardlayout.e-control .e-panel .e-panel-container .e-resize.e-double{
color: #0378d5;
}
`

```

```
font-size: 30px;
```

```
height: 20px;
```

```
width: 20px;
```

```
}
```

```
,
```

Customizing the dashboard layout panel background

Use the following CSS to customize the dashboard layout panel background.

```
`css
```

```
.e-dashboardlayout.e-control.e-responsive {
```

```
background: #b3d3ed;
```

```
}
```

```
,
```

You can refer to our [Angular Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Dashboard Layout example](#) to know how to present and manipulate data.

Accessibility in Angular Dashboard Layout component

The Dashboard Layout component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Dashboard Layout component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | Not applicable |

| [Accessibility Checker](#) Validation | |


```
| Axe-core Accessibility Validation |  |
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>
<div> - All
features of the component meet the requirement.</div>
<div> - Some features of the component do not meet the requirement.</div>
<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Dashboard Layout component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Dashboard Layout component:

| Attributes | Purpose |

| --- | --- |

| **role=list** | Indicates the role as a list for the Dashboard Layout element. |

| **role=listitem** | Indicates the role as a listitem for the Dashboard panels. |

| **role=presentation** | Indicates the role as a presentation for the table when the **showGridLines** property is enabled. |

| **aria-grabbed** | When the panel is chosen for dragging, the aria-grabbed attribute is set to "true". If it's set to "false", the element can be grabbed for drag-and-drop, but it won't be actively held. |

Keyboard interaction

Keyboard support is not applicable for the Dashboard Layout.

Ensuring accessibility

The Dashboard Layout component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Dashboard Layout component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Dashboard Layout component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

How To

Initializing dashboard using systemjs in Angular Dashboard layout component
DashboardLayout can also be initialized using **SystemJS** as follows.

Installation and configuration

- To setup basic **Angular** sample use the following commands.

`

```
git clone https://github.com/angular/quickstart.git quickstart
```

```
cd quickstart
```

```
npm install
```

`

For more information, refer to [Angular sample setup](#).

- Install Syncfusion DashboardLayout package using below command.

`

```
npm install @syncfusion/ej2-angular-layouts --save
```

`

The above package installs [Dashboard Layout component dependencies](#) which are required to render the component in an Angular environment.

- Syncfusion [ej2-angular-layouts](#) packages need to be mapped in **systemjs.config.js** configuration file.

```
`javascript
```

```
/
```

- System configuration for Angular samples
- Adjust as necessary for your application needs.

```
*/
```

```
(function (global) {
```

```
System.config({
```

```
paths: {
```

```
// paths serve as alias
```

```
'npm:': 'node_modules/',
```

```
"syncfusion:": "node_modules/@syncfusion/", // syncfusion alias
```

```
},
```

```
// map tells the System loader where to look for things
map: {
// our app is within the app folder
'app': 'app',
// angular bundles
'@angular/core': 'npm:@angular/core/bundles/core.umd.js',
'@angular/common': 'npm:@angular/common/bundles/common.umd.js',
'@angular/compiler': 'npm:@angular/compiler/bundles/compiler.umd.js',
'@angular/platform-browser': 'npm:@angular/platform-browser/bundles/platform-browser.umd.js',
'@angular/platform-browser-dynamic': 'npm:@angular/platform-browser-dynamic/bundles/platform-browser-dynamic.umd.js',
'@angular/http': 'npm:@angular/http/bundles/http.umd.js',
'@angular/router': 'npm:@angular/router/bundles/router.umd.js',
'@angular/forms': 'npm:@angular/forms/bundles/forms.umd.js',
// syncfusion bundles
"@syncfusion/ej2-base": "syncfusion:ej2-base/dist/ej2-base.umd.min.js",
"@syncfusion/ej2-layouts": "syncfusion:ej2-layouts/dist/ej2-layouts.umd.min.js",
"@syncfusion/ej2-angular-layouts": "syncfusion:ej2-angular-layouts/dist/ej2-angular-layouts.umd.min.js",
"@syncfusion/ej2-angular-base": "syncfusion:ej2-angular-base/dist/ej2-angular-base.umd.min.js",
// other libraries
'rxjs': 'npm:rxjs',
'angular-in-memory-web-api': 'npm:angular-in-memory-web-api/bundles/in-memory-web-api.umd.js'
},
// packages tells the System loader how to load when no filename and/or no extension
packages: {
app: {
defaultExtension: 'js',
meta: {
'./*.js': {
loader: 'systemjs-angular-loader.js'
}
}
},
},
},
```

```

rxjs: {
  defaultExtension: 'js'
}
}
});
})(this);
`

```

Adding style sheet to the application

To render the DashboardLayout component, import the DashboardLayout and its dependent component's styles as given below in `[src/styles.css]`.

```

`css
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-angular-layouts/styles/material.css";
`

```

Note: To refer the combined component styles, use Syncfusion [CRG](#) (Custom Resource Generator) in your application.

Add DashboardLayout to the application

You can render the DashboardLayout component by adding the panels property as the attribute to the HTML element directly. Add the HTML div element with panel definition for DashboardLayout into your `app.template.html` file.

`[src/app/app.template.html]`

```

`html
<div class="control-section">
  <ejs-dashboardlayout id='defaultLayout' [columns]="5" #defaultLayout [cellSpacing]='cellSpacing'>
    <div id="one" class="e-panel" data-row="0" data-col="0" data-sizeX="1" data-sizeY="1">
      <span id="close" class="e-template-icon e-clear-icon"></span>
      <div class="e-panel-container">
        <div class="text-align">0</div>
      </div>
    </div>
    <div id="two" class="e-panel" data-row="1" data-col="0" data-sizeX="1" data-sizeY="2">
      <span id="close" class="e-template-icon e-clear-icon"></span>
      <div class="e-panel-container">
        <div class="text-align">1</div>
      </div>
    </div>
  </ejs-dashboardlayout>
</div>
`

```

```

</div>
<div id="three" class="e-panel" data-row="0" data-col="1" data-sizeX="2" data-sizeY="2">
<span id="close" class="e-template-icon e-clear-icon"></span>
<div class="e-panel-container">
<div class="text-align">2</div>
</div>
</div>
<div id="four" class="e-panel" data-row="2" data-col="1" data-sizeX="1" data-sizeY="1">
<span id="close" class="e-template-icon e-clear-icon"></span>
<div class="e-panel-container">
<div class="text-align">3</div>
</div>
</div>
<div id="five" class="e-panel" data-row="2" data-col="2" data-sizeX="2" data-sizeY="1">
<span id="close" class="e-template-icon e-clear-icon"></span>
<div class="e-panel-container">
<div class="text-align">4</div>
</div>
</div>
<div id="six" class="e-panel" data-row="0" data-col="3" data-sizeX="1" data-sizeY="1">
<span id="close" class="e-template-icon e-clear-icon"></span>
<div class="e-panel-container">
<div class="text-align">5</div>
</div>
</div>
<div id="seven" class="e-panel" data-row="1" data-col="3" data-sizeX="1" data-sizeY="1">
<span id="close" class="e-template-icon e-clear-icon"></span>
<div class="e-panel-container">
<div class="text-align">6</div>
</div>
</div>
<div id="eight" class="e-panel" data-row="0" data-col="4" data-sizeX="1" data-sizeY="3">
<span id="close" class="e-template-icon e-clear-icon"></span>

```

```

<div class="e-panel-container">
<div class="text-align">7</div>
</div>
</div>
</ejs-dashboardlayout>
</div>
,

```

Now, modify the `templateUrl` in `app.component.ts` file to render DashboardLayout component.

[src/app/app.component.ts]

```

`typescript
import { Component } from '@angular/core';
@Component({
  selector: 'app-container',
  styleUrls: ['default-style.css'],
  templateUrl: 'app.template.html'
})
export class AppComponent {
}
,

```

- Import DashboardLayout module into Angular application(app.module.ts) from the package `@syncfusion/ej2-angular-layouts`.

```

`javascript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// import the DashboardLayoutModule for the Dashboard Layout component
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts';
import { AppComponent } from './app.component';
@NgModule({
  //declaration of ej2-angular-layouts module into NgModule
  imports: [ BrowserModule, DashboardLayoutModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]

```

```

})
export class AppModule { }
`

```

Run the application

Now, use the `npm start` command to run the application in the browser.

```

`html
npm start
`

```

The following example shows a basic DashboardLayout by adding the panels property directly into the HTML element.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DashboardLayoutModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [ DashboardLayoutModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./default-style.css'],
  template: `
    <div class="control-section">
      <ejs-dashboardlayout id='defaultLayout' #defaultLayout [columns]="5"
[cellSpacing]='cellSpacing'>
        <div id="one" class="e-panel" data-row="0" data-col="0" data-
sizeX="1" data-sizeY="1">
          <span id="close" class="e-template-icon e-clear-icon"></span>
          <div class="e-panel-container">
            <div class="text-align">0</div>
          </div>
        </div>
        <div id="two" class="e-panel" data-row="1" data-col="0" data-
sizeX="1" data-sizeY="2">
          <span id="close" class="e-template-icon e-clear-icon"></span>
          <div class="e-panel-container">
            <div class="text-align">1</div>
          </div>
        </div>
        <div id="three" class="e-panel" data-row="0" data-col="1" data-
sizeX="2" data-sizeY="2">
          <span id="close" class="e-template-icon e-clear-icon"></span>
          <div class="e-panel-container">
            <div class="text-align">2</div>
          </div>
        </div>
        <div id="four" class="e-panel" data-row="2" data-col="1" data-
sizeX="1" data-sizeY="1">
          <span id="close" class="e-template-icon e-clear-icon"></span>
          <div class="e-panel-container">
            <div class="text-align">3</div>
          </div>
        </div>
      </ejs-dashboardlayout>
    </div>
  `

```

```

        </div>
      </div>
      <div id="five" class="e-panel" data-row="2" data-col="2" data-
sizeX="2" data-sizeY="1">
        <span id="close" class="e-template-icon e-clear-icon"></span>
        <div class="e-panel-container">
          <div class="text-align">4</div>
        </div>
      </div>
      <div id="six" class="e-panel" data-row="0" data-col="3" data-
sizeX="1" data-sizeY="1">
        <span id="close" class="e-template-icon e-clear-icon"></span>
        <div class="e-panel-container">
          <div class="text-align">5</div>
        </div>
      </div>
      <div id="seven" class="e-panel" data-row="1" data-col="3" data-
sizeX="1" data-sizeY="1">
        <span id="close" class="e-template-icon e-clear-icon"></span>
        <div class="e-panel-container">
          <div class="text-align">6</div>
        </div>
      </div>
      <div id="eight" class="e-panel" data-row="0" data-col="4" data-
sizeX="1" data-sizeY="3">
        <span id="close" class="e-template-icon e-clear-icon"></span>
        <div class="e-panel-container">
          <div class="text-align">7</div>
        </div>
      </div>
    </ejs-dashboardlayout>
  </div>`,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  public cellSpacing: number[] = [10, 10];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

You can refer to our [Angular Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Dashboard Layout example](#) to know how to present and manipulate data.

DataManager

Getting started with Angular Data component

Dependencies

Below is the list of minimum dependencies required to use the DataManager.


```
`javascript
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-base
|-- es6-promise (Required when window.Promise is not available)
`
```

@syncfusion/ej2-data requires the presence of a Promise feature in global environment. In the browser, window.Promise must be available.

Setup Angular Environment

You can use [Angular CLI](#) to setup your Angular applications.

To install Angular CLI use the following command.

```
`bash
npm install -g @angular/cli@16.0.1
`
```

Create an Angular Application

Start a new Angular application using below Angular CLI command.

```
`bash
ng new data-app
`
```

This command will prompt you for a few settings for the new project, such as whether to add Angular routing and which stylesheet format to use.

```
D:\sample>ng new data-app
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
  SCSS  [ https://sass-lang.com/documentation/syntax#scss ]
  Sass  [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
  Less  [ http://lesscss.org ]
```

By default, it will create a CSS-based application.

Next, navigate to the created project folder:

```
`
cd data-app
`
```

Installing Syncfusion Data package

Syncfusion packages are distributed in npm as **@syncfusion** scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(>=20.2.36) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-data](#) package to the application.

```
`bash
npm install @syncfusion/ej2-data --save
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the [ngcc](#) package use the below.

Add [@syncfusion/ej2-angular-grids@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-grids@ngcc --save
```

To mention the ngcc package in the [package.json](#) file, add the suffix [-ngcc](#) with the package version as below.

```
`bash
@syncfusion/ej2-angular-grids:"20.2.38-ngcc"
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Connection to a data source

The DataManager can act as gateway for both local and remote data source which will uses the query to interact with the data source.

Binding to JSON data

DataManager can be bound to local data source by assigning the array of JavaScript objects to the **json** property or simply passing them to the constructor while instantiating.

Create a [src/app/datasource.ts] file and utilize the following dataset to provide JSON data.

```
`typescript
export let data: Object[] = [
{
  OrderID: 10248, CustomerID: 'VINET', EmployeeID: 5, OrderDate: new Date(8364186e5),
  ShipName: 'Vins et alcools Chevalier', ShipCity: 'Reims', ShipAddress: '59 rue de l Abbaye',
```

```
ShipRegion: 'CJ', ShipPostalCode: '51100', ShipCountry: 'France', Freight: 32.38, Verified: !0
},
{
  OrderID: 10249, CustomerID: 'TOMSP', EmployeeID: 6, OrderDate: new Date(836505e6),
  ShipName: 'Toms Spezialitäten', ShipCity: 'Münster', ShipAddress: 'Luisenstr. 48',
  ShipRegion: 'CJ', ShipPostalCode: '44087', ShipCountry: 'Germany', Freight: 11.61, Verified: !1
},
{
  OrderID: 10250, CustomerID: 'HANAR', EmployeeID: 4, OrderDate: new Date(8367642e5),
  ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress: 'Rua do Paço, 67',
  ShipRegion: 'RJ', ShipPostalCode: '05454-876', ShipCountry: 'Brazil', Freight: 65.83, Verified: !0
},
{
  OrderID: 10251, CustomerID: 'VICTE', EmployeeID: 3, OrderDate: new Date(8367642e5),
  ShipName: 'Victuailles en stock', ShipCity: 'Lyon', ShipAddress: '2, rue du Commerce',
  ShipRegion: 'CJ', ShipPostalCode: '69004', ShipCountry: 'France', Freight: 41.34, Verified: !0
},
{
  OrderID: 10252, CustomerID: 'SUPRD', EmployeeID: 4, OrderDate: new Date(8368506e5),
  ShipName: 'Suprêmes délices', ShipCity: 'Charleroi', ShipAddress: 'Boulevard Tirou, 255',
  ShipRegion: 'CJ', ShipPostalCode: 'B-6000', ShipCountry: 'Belgium', Freight: 51.3, Verified: !0
},
{
  OrderID: 10253, CustomerID: 'HANAR', EmployeeID: 3, OrderDate: new Date(836937e6),
  ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress: 'Rua do Paço, 67',
  ShipRegion: 'RJ', ShipPostalCode: '05454-876', ShipCountry: 'Brazil', Freight: 58.17, Verified: !0
},
{
  OrderID: 10254, CustomerID: 'CHOPS', EmployeeID: 5, OrderDate: new Date(8370234e5),
  ShipName: 'Chop-suey Chinese', ShipCity: 'Bern', ShipAddress: 'Hauptstr. 31',
  ShipRegion: 'CJ', ShipPostalCode: '3012', ShipCountry: 'Switzerland', Freight: 22.98, Verified: !1
},
{
```

```
OrderID: 10255, CustomerID: 'RICSU', EmployeeID: 9, OrderDate: new Date(8371098e5),
ShipName: 'Richter Supermarkt', ShipCity: 'Genève', ShipAddress: 'Starenweg 5',
ShipRegion: 'CJ', ShipPostalCode: '1204', ShipCountry: 'Switzerland', Freight: 148.33, Verified: !0
},
{
OrderID: 10256, CustomerID: 'WELLI', EmployeeID: 3, OrderDate: new Date(837369e6),
ShipName: 'Wellington Importadora', ShipCity: 'Resende', ShipAddress: 'Rua do Mercado, 12',
ShipRegion: 'SP', ShipPostalCode: '08737-363', ShipCountry: 'Brazil', Freight: 13.97, Verified: !1
},
{
OrderID: 10257, CustomerID: 'HILAA', EmployeeID: 4, OrderDate: new Date(8374554e5),
ShipName: 'HILARION-Abastos', ShipCity: 'San Cristóbal', ShipAddress: 'Carrera 22 con Ave. Carlos
Soubllette #8-35',
ShipRegion: 'Táchira', ShipPostalCode: '5022', ShipCountry: 'Venezuela', Freight: 81.91, Verified: !0
},
{
OrderID: 10258, CustomerID: 'ERNSH', EmployeeID: 1, OrderDate: new Date(8375418e5),
ShipName: 'Ernst Handel', ShipCity: 'Graz', ShipAddress: 'Kirchgasse 6',
ShipRegion: 'CJ', ShipPostalCode: '8010', ShipCountry: 'Austria', Freight: 140.51, Verified: !0
},
{
OrderID: 10259, CustomerID: 'CENTC', EmployeeID: 4, OrderDate: new Date(8376282e5),
ShipName: 'Centro comercial Moctezuma', ShipCity: 'México D.F.', ShipAddress: 'Sierras de Granada
9993',
ShipRegion: 'CJ', ShipPostalCode: '05022', ShipCountry: 'Mexico', Freight: 3.25, Verified: !1
},
{
OrderID: 10260, CustomerID: 'OTTIK', EmployeeID: 4, OrderDate: new Date(8377146e5),
ShipName: 'Ottilies Käseladen', ShipCity: 'Köln', ShipAddress: 'Mehrheimerstr. 369',
ShipRegion: 'CJ', ShipPostalCode: '50739', ShipCountry: 'Germany', Freight: 55.09, Verified: !0
},
{
OrderID: 10261, CustomerID: 'QUEDE', EmployeeID: 4, OrderDate: new Date(8377146e5),
ShipName: 'Que Delícia', ShipCity: 'Rio de Janeiro', ShipAddress: 'Rua da Panificadora, 12',
```

```

ShipRegion: 'RJ', ShipPostalCode: '02389-673', ShipCountry: 'Brazil', Freight: 3.05, Verified: !1
},
{
  OrderID: 10262, CustomerID: 'RATTC', EmployeeID: 8, OrderDate: new Date(8379738e5),
  ShipName: 'Rattlesnake Canyon Grocery', ShipCity: 'Albuquerque', ShipAddress: '2817 Milton Dr.',
  ShipRegion: 'NM', ShipPostalCode: '87110', ShipCountry: 'USA', Freight: 48.29, Verified: !0
}];
`

```

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { data } from '../datasource';
import { DataManager, Query } from '@syncfusion/ej2-data';
import { CommonModule } from '@angular/common';
@Component({
  imports: [CommonModule],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styles: [
    `
      .e-table {
        border: solid 1px #e0e0e0;
        border-collapse: collapse;
        font-family: Roboto;
      }
      .e-table td, .e-table th {
        border-style: solid;
        border-width: 1px 0 0;
        border-color: #e0e0e0;
        display: table-cell;
        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
      }
    `
  ]
})
export class AppComponent implements OnInit {
  public items?: object[] | any;
  public ngOnInit(): void {
    this.items = new DataManager(data as JSON[]).executeLocal(new
    Query().take(6));
  }
}

```

APP.COMPONENT.HTML

```
{% raw %}
<table class='e-table'>
<tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
<tr *ngFor="let item of items">
<td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
/td>
</tr>
</table>
{% endraw %}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Binding to OData

DataManager can be bound to remote data source by assigning service end point URL to the **url** property.

Now all **DataManager** operations will address the provided service end point.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ReturnOption } from '@syncfusion/ej2-data';
import { CommonModule } from '@angular/common';
const SERVICE_URI =
'https://services.syncfusion.com/angular/production/api/Orders';
@Component({
imports: [CommonModule],
standalone: true,
selector: 'app-root',
templateUrl: './app.component.html',
styles: [
.e-table {
border: solid 1px #e0e0e0;
border-collapse: collapse;
font-family: Roboto;
}
.e-table td, .e-table th {
border-style: solid;
border-width: 1px 0 0;
border-color: #e0e0e0;
display: table-cell;
font-size: 14px;
line-height: 20px;
overflow: hidden;
padding: 8px 21px;
vertical-align: middle;
white-space: nowrap;
```

```

        width: auto;
    }
    `]
})
export class AppComponent implements OnInit {
    public items?: object[] | any;
    public ngOnInit(): void {
        new DataManager({ url: SERVICE_URI }).executeQuery(new
        Query()).then((e: ReturnOption) => {
            this.items = e.result as object[];
        }).catch((e) => true);
    }
}

```

APP.COMPONENT.HTML

```

{% raw %}
<table class='e-table'>
<tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
<tr *ngFor="let item of items">
<td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
/td>
</tr>
</table>
{% endraw %}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Filter

The data filtering is a trivial operation which will let us to get reduced view of data based on filter criteria.

The filter expression can be built easily using **where** method of **Query** class.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { data } from './datasource';
import { DataManager, Query } from '@syncfusion/ej2-data';
import { CommonModule } from '@angular/common';
@Component({
    imports: [CommonModule],
    standalone: true,
    selector: 'app-root',
    templateUrl: './app.component.html',
    styles: [
        .e-table {
            border: solid 1px #e0e0e0;

```

```

        border-collapse: collapse;
        font-family: Roboto;
    }
    .e-table td, .e-table th {
        border-style: solid;
        border-width: 1px 0 0;
        border-color: #e0e0e0;
        display: table-cell;
        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
    }
}
` ]
})
export class AppComponent implements OnInit {
    public items?: object[] | any;
    public ngOnInit(): void {
        this.items = new DataManager(data as JSON[]).executeLocal(new
Query().where('EmployeeID', 'equal', 3));
    }
}

```

APP.COMPONENT.HTML

```

{% raw %}
<table class='e-table'>
<tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
<tr *ngFor="let item of items">
<td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
/td>
</tr>
</table>
{% endraw %}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Sort

The data can be ordered either in ascending or descending using **sortBy** method of **Query** class.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CommonModule } from '@angular/common';
import { Component, OnInit } from '@angular/core';
import { data } from './datasource';

```



```

import { DataManager, Query } from '@syncfusion/ej2-data';
@Component({
  imports: [CommonModule],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styles: [
    .e-table {
      border: solid 1px #e0e0e0;
      border-collapse: collapse;
      font-family: Roboto;
    }
    .e-table td, .e-table th {
      border-style: solid;
      border-width: 1px 0 0;
      border-color: #e0e0e0;
      display: table-cell;
      font-size: 14px;
      line-height: 20px;
      overflow: hidden;
      padding: 8px 21px;
      vertical-align: middle;
      white-space: nowrap;
      width: auto;
    }
  ]
})
export class AppComponent implements OnInit {
  public items?: object[] | any;
  public ngOnInit(): void {
    this.items = new DataManager(data as JSON[]).executeLocal(new
    Query().sortBy('CustomerID').take(8));
  }
}

```

APP.COMPONENT.HTML

```

{% raw %}
<table class='e-table'>
<tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
<tr *ngFor="let item of items">
<td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
/td>
</tr>
</table>
{% endraw %}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Page

The **page** method of the Query class is used to get range of data based on the page number and the total page size.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CommonModule } from '@angular/common';
import { Component, OnInit } from '@angular/core';
import { data } from '../datasource';
import { DataManager, Query } from '@syncfusion/ej2-data';
@Component({
  imports: [CommonModule],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styles: [
    `
      .e-table {
        border: solid 1px #e0e0e0;
        border-collapse: collapse;
        font-family: Roboto;
      }
      .e-table td, .e-table th {
        border-style: solid;
        border-width: 1px 0 0;
        border-color: #e0e0e0;
        display: table-cell;
        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
      }
    `
  ]
})
export class AppComponent implements OnInit {
  public items?: object[] | any;
  public ngOnInit(): void {
    this.items = new DataManager(data as JSON[]).executeLocal(new
    Query().page(1, 8));
  }
}
```

APP.COMPONENT.HTML

```
{% raw %}
<table class='e-table'>
<tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
<tr *ngFor="let item of items">
<td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
/td>
</tr>
</table>
```

```
{% endraw %}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Component binding

DataManager component can be used with Syncfusion components which supports data binding.

In the following samples, the grid component is bound. To render the grid with the necessary configurations, please refer to the [Grid Getting Started](#) documentation.

Local data binding

A DataSource can be created in-line with other Syncfusion component configuration settings.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { GridModule } from '@syncfusion/ej2-angular-grids'
import { Component, OnInit } from '@angular/core';
import { data } from './datasource';
import { DataManager } from '@syncfusion/ej2-data';
@Component({
  imports: [
    GridModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-grid [dataSource]='data' height="315px">
    <e-columns>
      <e-column field='OrderID' headerText='Order ID'
textAlign='Right' width=90></e-column>
      <e-column field='CustomerID' headerText='Customer ID'
width=120></e-column>
      <e-column field='Freight' headerText='Freight'
textAlign='Right' format='C2' width=90></e-column>
      <e-column field='OrderDate' headerText='Order Date'
textAlign='Right' format='yMd' width=120></e-column>
    </e-columns>
  </ejs-grid>`
})
export class AppComponent implements OnInit {
  public data?: DataManager;
  public ngOnInit(): void {
    this.data = new DataManager(data as JSON[]);
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Remote data binding

To bind remote data to Syncfusion component, you can assign a service data as an instance of **DataManager** to the [dataSource](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { GridModule } from '@syncfusion/ej2-angular-grids'
import { Component, OnInit } from '@angular/core';
import { DataManager } from '@syncfusion/ej2-data';
const SERVICE_URI = 'https://services.syncfusion.com/angular/production/';
@Component({
  imports: [
    GridModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-grid [dataSource]='data' height="315px">
    <e-columns>
      <e-column field='OrderID' headerText='Order ID'
textAlign='Right' width=90></e-column>
      <e-column field='CustomerID' headerText='Customer ID'
width=120></e-column>
      <e-column field='EmployeeID' headerText='EmployeeID'
textAlign='Right' width=90></e-column>
      <e-column field='ShipName' headerText='Ship Name'
textAlign='Left' width=120></e-column>
    </e-columns>
  </ejs-grid>`
})
export class AppComponent implements OnInit {
  public data?: DataManager;
  public ngOnInit(): void {
    this.data = new DataManager({ url: SERVICE_URI+ 'api/Orders' });
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Data binding in Angular Data component

[DataManager](#) supports both RESTful JSON data services binding and local JavaScript object array binding.

Local data binding

[DataManager](#) can be bound to local datasource by assigning the array of JavaScript objects to the **json** property or simply passing them to the constructor while instantiating. Now the JavaScript object array can be queried and manipulated.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CommonModule } from '@angular/common';
import { Component, OnInit } from '@angular/core';
import { data } from './datasource';
import { DataManager, Query } from '@syncfusion/ej2-data';
@Component({
  imports: [CommonModule],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',
  styles: [
    `
    .e-table {
      border: solid 1px #e0e0e0;
      border-collapse: collapse;
      font-family: Roboto;
    }
    .e-table td, .e-table th {
      border-style: solid;
      border-width: 1px 0 0;
      border-color: #e0e0e0;
      display: table-cell;
      font-size: 14px;
      line-height: 20px;
      overflow: hidden;
      padding: 8px 21px;
      vertical-align: middle;
      white-space: nowrap;
      width: auto;
    }
    `
  ]
})
export class AppComponent implements OnInit {
  public items?: object[] | any;
  public ngOnInit(): void {
    this.items = new DataManager(data as JSON[]).executeLocal(new
    Query().take(8));
  }
}
```

APP.TEMPLATE.HTML

```
<table class='e-table'>
  <tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
  <tr *ngFor="let item of items">

    <td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
    /td>
  </tr>
```

```
</table>
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Remote data binding

[DataManager](#) can be bound to remote datasource by assigning service end point URL to the **url** property. With the provided **url**, the [DataManager](#) handles all communication with the data server with help of queries.

When querying data, the [DataManager](#) will convert the query object([Query](#)) into server request after calling [executeQuery](#) and waits for the server response(**JSON** format).

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ReturnOption } from '@syncfusion/ej2-data';
const SERVICE_URI =
'https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Orders';
@Component({
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',
  styles: [
    .e-table {
      border: solid 1px #e0e0e0;
      border-collapse: collapse;
      font-family: Roboto;
    }
    .e-table td, .e-table th {
      border-style: solid;
      border-width: 1px 0 0;
      border-color: #e0e0e0;
      display: table-cell;
      font-size: 14px;
      line-height: 20px;
      overflow: hidden;
      padding: 8px 21px;
      vertical-align: middle;
      white-space: nowrap;
      width: auto;
    }
  ]
})
export class AppComponent implements OnInit {
  public items?: object[] | any;
  public ngOnInit(): void {
    new DataManager({ url: SERVICE_URI }).executeQuery(new
    Query().take(6)).then((e: ReturnOption) => {
```

```

        this.items = e.result as object[];
    }).catch((e) => true);
}
}

```

APP.TEMPLATE.HTML

```

<table class='e-table'>
  <tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
  <tr *ngFor="let item of items">

<td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
/td>
  </tr>
</table>

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

The queried data will not be cached locally unless offline mode is enabled.

See Also

- [Binding with OData service](#)
- [Binding with ODataV4 service](#)
- [Binding with Web API](#)
- [How to write custom adaptor](#)
- [How to work in offline mode](#)
- [How to send additional parameters](#)
- [How to add custom request headers](#)

Adaptors in Angular Data component

Each datasource or remote service uses different way in accepting request and sending back the response. [DataManager](#) cannot anticipate every way a datasource works. To tackle this problem the [DataManager](#) uses the adaptor concept to communicate with particular data source.

For local datasources, the role of the data adaptor is to query the JavaScript object array based on the [Query](#) object and manipulate them.

When comes with remote datasource, the data adaptor is used to send the request that the server can understand and process the server response.

The adaptor can be assigned using the **adaptor** property of the [DataManager](#).

Json adaptor

JsonAdaptor is used to query and manipulate JavaScript object array.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { data } from './datasource';
import { DataManager, Query, JsonAdaptor } from '@syncfusion/ej2-data';
import { CommonModule } from '@angular/common';
@Component({
  imports: [CommonModule],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',
  styles: [
    .e-table {
      border: solid 1px #e0e0e0;
      border-collapse: collapse;
      font-family: Roboto;
    }
    .e-table td, .e-table th {
      border-style: solid;
      border-width: 1px 0 0;
      border-color: #e0e0e0;
      display: table-cell;
      font-size: 14px;
      line-height: 20px;
      overflow: hidden;
      padding: 8px 21px;
      vertical-align: middle;
      white-space: nowrap;
      width: auto;
    }
  ]
})
export class AppComponent implements OnInit {
  public items?: object[] | any;
  public ngOnInit(): void {
    this.items = new DataManager({ json: data, adaptor: new JsonAdaptor()
  }).executeLocal(new Query().take(8));
  }
}

```

APP.TEMPLATE.HTML

```

<table class='e-table'>
  <tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
  <tr *ngFor="let item of items">

<td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
/td>
  </tr>
</table>

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';

```



```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Url adaptor

UrlAdaptor act as the base adaptor for interacting with remote data services. Most of the built-in adaptors are derived from the **UrlAdaptor**.

`typescript

```
import { DataManager, Query, UrlAdaptor } from '@syncfusion/ej2-data';
const SERVICE_URI = 'https://ej2services.syncfusion.com/angular/development/api/UrlDataSource';
new DataManager({
  url: SERVICE_URI,
  adaptor: new UrlAdaptor()
}).executeQuery(new Query().take(8)).then((e) => {
  //e.result will contain the records
});
`
```

UrlAdaptor expects response as a JSON object with properties **result** and **count** which contains the collection of entities and the total number of records respectively.

The sample response object should be as follows,

`json

```
{
  "result": [{..}, {..}, {..}, ...],
  "count": 67
}
```

OData adaptor

OData is standardized protocol for creating and consuming data. You can retrieve data from OData service using [DataManager](#). The **ODataAdaptor** helps you to interact with OData service. You can refer to the following code example of remote Data binding using OData service.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ODataAdaptor, ReturnOption } from
 '@syncfusion/ej2-data';
const SERVICE_URI =
 'https://services.odata.org/V3/Northwind/Northwind.svc/Orders/';
@Component({
  standalone: true,
  selector: 'app-root',
```

```

    templateUrl: './app.template.html',
    styles: [
      .e-table {
        border: solid 1px #e0e0e0;
        border-collapse: collapse;
        font-family: Roboto;
      }
      .e-table td, .e-table th {
        border-style: solid;
        border-width: 1px 0 0;
        border-color: #e0e0e0;
        display: table-cell;
        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
      }
    ]
  })
  export class AppComponent implements OnInit {
    public items?: object[] | any;
    public ngOnInit(): void {
      new DataManager({ url: SERVICE_URI, adaptor: new ODataAdaptor() })
        .executeQuery(new Query().take(8)).then((e: ReturnOption) =>
        this.items = e.result as object[]).catch((e) => true);
    }
  }
}

```

APP.TEMPLATE.HTML

```

<table class='e-table'>
  <tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
  <tr *ngFor="let item of items">

    <td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
    /td>
    </tr>
</table>

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

By default, **ODataAdaptor** is used by [DataManager](#).

ODataV4 adaptor

The ODataV4 is an improved version of OData protocols and the [DataManager](#) can also retrieve and consume OData v4 services. For more details on OData v4 Services, refer the [odata documentation](#).

You can use the **ODataV4Adaptor** to interact with ODataV4 service.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CommonModule } from '@angular/common';
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ODataV4Adaptor, ReturnOption } from
 '@syncfusion/ej2-data';
const SERVICE_URI =
 'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
@Component({
  imports: [CommonModule],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',
  styles: [
    .e-table {
      border: solid 1px #e0e0e0;
      border-collapse: collapse;
      font-family: Roboto;
    }
    .e-table td, .e-table th {
      border-style: solid;
      border-width: 1px 0 0;
      border-color: #e0e0e0;
      display: table-cell;
      font-size: 14px;
      line-height: 20px;
      overflow: hidden;
      padding: 8px 21px;
      vertical-align: middle;
      white-space: nowrap;
      width: auto;
    }
  ]
})
export class AppComponent implements OnInit {
  public items?: object[] | any;
  public ngOnInit(): void {
    new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor() })
      .executeQuery(new Query().take(8)).then((e: ReturnOption) =>
this.items = e.result as object[]).catch((e) => true);
  }
}
```

APP.TEMPLATE.HTML

```
<table class='e-table'>
  <tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
  <tr *ngFor="let item of items">

<td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
/td>
  </tr>
</table>
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Web API adaptor

You can use the **WebApiAdaptor** to interact with Web API created with OData endpoint. The **WebApiAdaptor** is extended from the **ODataAdaptor**. Hence to use **WebApiAdaptor**, the endpoint should understand the OData formatted queries send along with request.

To enable OData query option for Web API, please refer to the [documentation](#)

```
`typescript
```

```
import { DataManager, Query, WebApiAdaptor } from '@syncfusion/ej2-data';
const SERVICE_URI = 'https://ej2services.syncfusion.com/angular/development/api/Orders';
new DataManager({
  url: SERVICE_URI,
  adaptor: new WebApiAdaptor()
}).executeQuery(new Query().take(8)).then((e) => {
  //e.result will contain the records
});
`
```

WebApiAdaptor expects JSON response from the server and the response object should contain properties **Items** and **Count** whose values are collection of entities and total count of the entities respectively.

The sample response object should look like below.

```
`json
{
  Items: [{..}, {..}, {..}, ...],
  Count: 830
}
`
```

WebMethod Adaptor

The **WebMethodAdaptor** is used to bind data source from remote services and code behind methods. It can be enabled in Grid using Adaptor property of DataManager as **WebMethodAdaptor**.

For every operations, an Fetch post will be send to the specified data service.

```
`typescript
```

```
import { DataManager, Query, WebMethodAdaptor } from '@syncfusion/ej2-data';
let SERVICE_URI = 'Default.aspx/DataSource';
new DataManager({
  url: SERVICE_URI,
  adaptor: new WebMethodAdaptor
}).executeQuery(new Query().take(8)).then((e) => {
  //e.result will contain the records
});
`
```

WebMethodAdaptor expects JSON response from the server and the response object should contain properties **result** and **count** whose values are collection of entities and total count of the entities respectively.

The sample response object should look like below.

```
`json
{
  result: [{..}, {..}, {..}, ...],
  count: 830
}
`
```

The controller method's data parameter name must be **value**.

GraphQL Adaptor

The **GraphQLAdaptor** provides an option to retrieve data from the GraphQL server. It performs CRUD and data operations such as paging, sorting, filtering etc by sending the required arguments to the server.

You can provide the GraphQL query string by using the **query** property of the **GraphQLAdaptor**. Since, the **GraphQLAdaptor** is extended from the **UrlAdaptor**, it expects response as a JSON object with properties **result** and **count** which contains the collection of entities and the total number of records respectively. The GraphQL response should be returned in JSON format like { "data": { ... } } with query name as field, you need to set the **result** and **count** properties to map the response.

```
`typescript
import { DataManager, Query, GraphQLAdaptor } from '@syncfusion/ej2-data';
const SERVICE_URI: string = 'http://controller.com/actions';
new DataManager({
  url: SERVICE_URI, adaptor: new GraphQLAdaptor({
    response: {
      result: 'getOrders.OrderData',
```

```
count: 'getOrders.OrderCount'
},
query: `query getOrders($datamanager: String) {
  getOrders(datamanager: $datamanager) {
    OrderCount,
    OrderData{OrderID, CustomerID, EmployeeID, ShipCity, ShipCountry}
  }
}`
}}
}).executeQuery(new Query().take(8)).then((e) => {
//e.result will contain the records
});
`
```

The Schema for the GraphQL server is

```
`typescript
input OrderInput {
  OrderID: Int!
  CustomerID: String!
  EmployeeID: Int!
  ShipCity: String!
  ShipCountry: String!
}
type Order {
  OrderID: Int!
  CustomerID: String!
  EmployeeID: Int!
  ShipCity: String!
  ShipCountry: String!
}
type ReturnType {
  getOrders: [Order]
  count: Int
}
```

```

type Query {
  getOrders(datamanager: String): ReturnType
}

type Mutation {
  createOrder(value: OrderInput): Order!
  updateOrder(key: Int!, keyColumn: String, value: OrderInput): Order
  deleteOrder(key: Int!, keyColumn: String, value: OrderInput): Order!
}
`

```

The resolver for the corresponding action is

```

`typescript
import { data } from "./db";

const resolvers = {
  Query: {
    getOrders: (parent, { datamanager }, context, info) => {
      if (datamanager.search) {
        // Perform searching
      }
      if (datamanager.sorted) {
        // Perform sorting
      }
      if (datamanager.where) {
        // Perform filtering
      }
      if (datamanager.search) {
        // Perform search
      }
      if (datamanager.skip && datamanager.take) {
        // Perform Paging
      }
      return { OrderData: data, OrderCount: data.length };
    },
  },
}

```

```

Mutation: {
  createOrder: (parent, { value }, context, info) => {
    // Perform Insert
    return value;
  },
  updateOrder: (parent, { key, keyColumn, value }, context, info) => {
    // Perform Update
    return value;
  },
  deleteOrder: (parent, { key, keyColumn, value }, context, info) => {
    // Perform Delete
    return value;
  },
};
export default resolvers;
`

```

The query parameters will be send in a string format which contains the below details.

Parameters	Description
RequiresCounts	If it is true then the total count of records will be included in response.
Skip	Holds the number of records to skip.
Take	Holds the number of records to take.
Sorted	Contains details about current sorted column and its direction.
Where	Contains details about current filter column name and its constraints.
Group	Contains details about current Grouped column names.

Performing CRUD action with GraphQLAdaptor

You can perform the CRUD actions by returning the mutation queries inside the `getMutation` method based on the action.

```

`typescript

```

```

import { DataManager, Query, GraphQLAdaptor } from '@syncfusion/ej2-data';

```

```

const SERVICE_URI: string = 'http://controller.com/actions';

```

```

new DataManager({

```



```
url: SERVICE_URI, adaptor: new GraphQLAdaptor({
  response: {
    result: 'getOrders.getOrders',
    count: 'getOrders.count'
  },
  query: `query getOrders($datamanager: String) {
    getOrders(datamanager: $datamanager) {
      count,
      getOrders{OrderID, CustomerID, EmployeeID, ShipCity, ShipCountry}
    }
  }`,
  getMutation: function (action): string {
    if (action === 'insert') {
      return `mutation CreateOrderMutation($value: OrderInput!){
        createOrder(value: $value){
          OrderID, CustomerID, EmployeeID, ShipCity, ShipCountry
        }
      }`;
    }
    if (action === 'update') {
      return `mutation Update($key: ID!, $keyColumn: String,$value: OrderInput){
        updateOrder(key: $key, keyColumn: $keyColumn, value: $value) {
          OrderID, CustomerID, EmployeeID, ShipCity, ShipCountry
        }
      }`;
    }
    } else {
      return `mutation Remove($key: ID!, $keyColumn: String, $value: OrderInput){
        deleteOrder(key: $key, keyColumn: $keyColumn, value: $value) {
          OrderID, CustomerID, EmployeeID, ShipCity, ShipCountry
        }
      }`;
    }
  }
})
```

```

}).executeQuery(new Query().take(8)).then((e) => {
//e.result will contain the records
});
`

```

Writing custom adaptor

Sometimes the built-in adaptors does not meet your requirement. In such cases you can create your own adaptor.

To create and use custom adaptor, please refer to the below steps.

- Select an built-in adaptor which will act as base class for your custom adaptor.
- Override the desired method to achieve your requirement.
- Assign the custom adaptor to the **adaptor** property of [DataManager](#).

For the sake of demonstrating custom adaptor approach, we are going to see how to add serial number for the records by overriding the built-in response processing using **processResponse** method of the **ODataAdaptor**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ODataAdaptor, ReturnOption } from
'@syncfusion/ej2-data';
import { CommonModule } from '@angular/common';
const SERVICE_URI =
'https://services.odata.org/V3/Northwind/Northwind.svc/Orders/';
class SerialNoAdaptor extends ODataAdaptor {
    public override processResponse(): object {
        let i: number = 0;
        //calling base class processResponse function
        let original: object[] | any = super.processResponse.apply(this,
arguments as any);
        //Adding serial number
        original.forEach((item: object | any) => item['Sno'] = ++i);
        return original;
    }
}
@Component({
imports: [CommonModule],
standalone: true,
selector: 'app-root',
templateUrl: './app.template.html',
styles: [
.e-table {
border: solid 1px #e0e0e0;
border-collapse: collapse;
font-family: Roboto;
}
.e-table td, .e-table th {
border-style: solid;

```

```

        border-width: 1px 0 0;
        border-color: #e0e0e0;
        display: table-cell;
        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
    }
    `]
})
export class AppComponent implements OnInit {
    public items?: object[] | any;
    public ngOnInit(): void {
        new DataManager({ url: SERVICE_URI, adaptor: new SerialNoAdaptor })
            .executeQuery(new Query().take(8)).then((e: ReturnOption) =>
        this.items = <object[]>e.result).catch((e) => true);
    }
}

```

APP.TEMPLATE.HTML

```

<table class='e-table'>
    <tr><th>SNO</th><th>Customer ID</th><th>Employee ID</th></tr>
    <tr *ngFor="let item of items">

<td>{{item.Sno}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}</td>
    </tr>
</table>

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Querying in Angular Data component

In this section, you will see in detail about how to build query using [Query](#) class and consume the data source.

Specifying resource name using `from`

The [from](#) method is used to specify the resource name or table name from where the data should be retrieved.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ODataAdaptor, ReturnOption } from
 '@syncfusion/ej2-data';

```

```

const SERVICE_URI =
'https://services.odata.org/V3/Northwind/Northwind.svc/Orders/';
@Component({
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',
  styles: [
    .e-table {
      border: solid 1px #e0e0e0;
      border-collapse: collapse;
      font-family: Roboto;
    }
    .e-table td, .e-table th {
      border-style: solid;
      border-width: 1px 0 0;
      border-color: #e0e0e0;
      display: table-cell;
      font-size: 14px;
      line-height: 20px;
      overflow: hidden;
      padding: 8px 21px;
      vertical-align: middle;
      white-space: nowrap;
      width: auto;
    }
  ]
})
export class AppComponent implements OnInit {
  public items?: object[] | any;
  public ngOnInit(): void {
    new DataManager({ url: SERVICE_URI, adaptor: new ODataAdaptor() })
      .executeQuery(new Query().from('Orders').take(8)).then((e:
ReturnOption) => this.items = e.result as object[]).catch((e) => true);
  }
}

```

APP.TEMPLATE.HTML

```

<table class='e-table'>
  <tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
  <tr *ngFor="let item of items">

    <td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
    /td>
  </tr>
</table>

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Projection using `select`

The [select](#) method is used to select particular fields or columns from the data source.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ODataAdaptor, ReturnOption } from
 '@syncfusion/ej2-data';
const SERVICE_URI =
 'https://services.odata.org/V3/Northwind/Northwind.svc/Orders/';
@Component({
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',
  styles: [ `
    .e-table {
      border: solid 1px #e0e0e0;
      border-collapse: collapse;
      font-family: Roboto;
    }
    .e-table td, .e-table th {
      border-style: solid;
      border-width: 1px 0 0;
      border-color: #e0e0e0;
      display: table-cell;
      font-size: 14px;
      line-height: 20px;
      overflow: hidden;
      padding: 8px 21px;
      vertical-align: middle;
      white-space: nowrap;
      width: auto;
    }
  ` ]
})
export class AppComponent implements OnInit {
  public items?: object[] | any;
  public ngOnInit(): void {
    new DataManager({ url: SERVICE_URI, adaptor: new ODataAdaptor() })
      .executeQuery(new Query().select(['OrderID', 'CustomerID',
 'EmployeeID']).take(8))
      .then((e: ReturnOption) => this.items = e.result as
 object[]).catch((e) => true);
  }
}
```

APP.TEMPLATE.HTML

```
<table class='e-table'>
  <tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
  <tr *ngFor="let item of items">

  <td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
  /td>
```

```
</tr>
</table>
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Eager loading navigation properties

You can use the [expand](#) method to eagerly load navigation properties. The navigation properties values are accessed using appropriate field names separated by dot(.) sign.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ReturnOption } from '@syncfusion/ej2-data';
const SERVICE_URI =
'https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Orders';
@Component({
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',
  styles: [`
    .e-table {
      border: solid 1px #e0e0e0;
      border-collapse: collapse;
      font-family: Roboto;
    }
    .e-table td, .e-table th {
      border-style: solid;
      border-width: 1px 0 0;
      border-color: #e0e0e0;
      display: table-cell;
      font-size: 14px;
      line-height: 20px;
      overflow: hidden;
      padding: 8px 21px;
      vertical-align: middle;
      white-space: nowrap;
      width: auto;
    }
  `]
})
export class AppComponent implements OnInit {
  public items?: object[] | any;
  public ngOnInit(): void {
    new DataManager({ url: SERVICE_URI })
      .executeQuery(new Query().expand('Employee').select(['OrderID',
'CustomerID', 'Employee.FirstName']).take(8))
      .then((e: ReturnOption) => this.items = e.result as
object[]).catch((e) => true);
```

```
}
}
```

APP.TEMPLATE.HTML

```
<table class='e-table'>
  <tr><th>Order ID</th><th>Customer ID</th><th>Employee Name</th></tr>
  <tr *ngFor="let item of items">

<td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.Employee.Firs
tName}}</td>
  </tr>
</table>
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Sorting

You can use the [sortBy](#) method to perform sort operation in the

data source. Default sorting order is **ascending**. To change the sort order, either you can

specify the second argument of [sortBy](#) as **descending** or use the

[sortByDesc](#) method.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ODataAdaptor, ReturnOption } from
'@syncfusion/ej2-data';
const SERVICE_URI: string =
'https://services.odata.org/V3/Northwind/Northwind.svc/Orders/';
@Component({
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',
  styles: [`
    .e-table {
      border: solid 1px #e0e0e0;
      border-collapse: collapse;
      font-family: Roboto;
    }
    .e-table td, .e-table th {
      border-style: solid;
      border-width: 1px 0 0;
      border-color: #e0e0e0;
      display: table-cell;
      font-size: 14px;
      line-height: 20px;
```

```

        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
    }
}
])
})
export class AppComponent implements OnInit {
    public items?: object[] | any;
    public ngOnInit(): void {
        new DataManager({ url: SERVICE_URI, adaptor: new ODataAdaptor})
            .executeQuery(new Query().sortBy('CustomerID', 'descending').take(8))
            .then((e: ReturnOption) => this.items = e.result as
object[]).catch((e) => true);
    }
}

```

APP.TEMPLATE.HTML

```

<table class='e-table'>
    <tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
    <tr *ngFor="let item of items">

<td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
/td>
    </tr>
</table>

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Multi sorting can be performed by simply chaining the multiple `sortBy` methods.

Filtering

You can use the [where](#) method to build filter criteria which allows you to get reduced view of records. The [where](#) method can also be chained to form multiple filter criteria.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ODataAdaptor, ReturnOption } from
'@syncfusion/ej2-data';
const SERVICE_URI =
'https://services.odata.org/V3/Northwind/Northwind.svc/Orders/';
@Component({
    standalone: true,
    selector: 'app-root',

```



```

    templateUrl: './app.template.html',
    styles: [
      .e-table {
        border: solid 1px #e0e0e0;
        border-collapse: collapse;
        font-family: Roboto;
      }
      .e-table td, .e-table th {
        border-style: solid;
        border-width: 1px 0 0;
        border-color: #e0e0e0;
        display: table-cell;
        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
      }
    ]
  })
  export class AppComponent implements OnInit {
    public items?: object[] | any;
    public ngOnInit(): void {
      new DataManager({ url: SERVICE_URI, adaptor: new ODataAdaptor() })
        .executeQuery(new Query().sortBy('CustomerID', 'descending').take(8))
        .then((e: ReturnOption) => this.items = e.result as
object[]).catch((e) => true);
    }
  }
}

```

APP.TEMPLATE.HTML

```

<table class='e-table'>
  <tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
  <tr *ngFor="let item of items">

<td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
/td>
  </tr>
</table>

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Filter Operators

Filter operators are generally used to specify the filter type. The various filter operators supported by [DataManager](#) is listed below.

- `greaterthan`
- `greaterthanorequal`
- `lessthan`
- `lessthanorequal`
- `equal`
- `notequal`
- `startswith`
- `endswith`
- `contains`

These filter operators are used for creating filter query using

[where](#) method and [Predicate](#) class.

Build complex filter criteria using `Predicate`

Sometimes chaining [where](#) method is not sufficient to create very

complex filter criteria, in such cases we can use [Predicate](#) class to create composite filter criteria.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ODataAdaptor, Predicate, ReturnOption } from
 '@syncfusion/ej2-data';
const SERVICE_URI =
 'https://services.odata.org/V3/Northwind/Northwind.svc/Orders/';
@Component({
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',
  styles: [
    `
      .e-table {
        border: solid 1px #e0e0e0;
        border-collapse: collapse;
        font-family: Roboto;
      }
      .e-table td, .e-table th {
        border-style: solid;
        border-width: 1px 0 0;
        border-color: #e0e0e0;
        display: table-cell;
        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
      }
    `
  ]
})
export class AppComponent implements OnInit {
  public items?: object[] | any;
  public ngOnInit(): void {
```

```

    let predicate: Predicate = new Predicate('EmployeeID', 'equal', 3);
    predicate = predicate.or('EmployeeID', 'equal', 2);
    new DataManager({ url: SERVICE_URI, adaptor: new ODataAdaptor() })
      .executeQuery(new Query().where(predicate).take(8))
      .then((e: ReturnOption) => this.items = e.result as
object[]).catch((e) => true);
  }
}

```

APP.TEMPLATE.HTML

```

<table class='e-table'>
  <tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
  <tr *ngFor="let item of items">

<td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
/td>
  </tr>
</table>

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Searching

You can use the [search](#) method to create search criteria, it

differs from the filter in the way that search criteria will applied to all fields in the data

source whereas filter criteria will be applied to a particular field.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ReturnOption } from '@syncfusion/ej2-data';
import { data } from './datasource';
@Component({
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',
  styles: [
    .e-table {
      border: solid 1px #e0e0e0;
      border-collapse: collapse;
      font-family: Roboto;
    }
    .e-table td, .e-table th {
      border-style: solid;
      border-width: 1px 0 0;
      border-color: #e0e0e0;
      display: table-cell;
    }
  ]
})

```

```

        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
    }
}
`]
})
export class AppComponent implements OnInit {
    public items?: object[] | any;
    public ngOnInit(): void {
        new DataManager(data as JSON[])
            .executeQuery(new Query().search('VI', ['CustomerID']))
            .then((e: ReturnOption) => this.items = e.result as
object[]).catch((e) => true);
    }
}

```

APP.TEMPLATE.HTML

```

<table class='e-table'>
    <tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
    <tr *ngFor="let item of items">

<td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
/td>
    </tr>
</table>

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

You can search particular fields by passing the field name collection in the second argument of [search](#) method.

Grouping

[DataManager](#) allow you to group records by category. The

[group](#) method is used to add group query.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ODataAdaptor, ReturnOption } from
'@syncfusion/ej2-data';
const SERVICE_URI =
'https://services.odata.org/V3/Northwind/Northwind.svc/Orders/';

```

```

@Component({
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',
  styles: [
    .e-table {
      border: solid 1px #e0e0e0;
      border-collapse: collapse;
      font-family: Roboto;
    }
    .e-table td, .e-table th {
      border-style: solid;
      border-width: 1px 0 0;
      border-color: #e0e0e0;
      display: table-cell;
      font-size: 14px;
      line-height: 20px;
      overflow: hidden;
      padding: 8px 21px;
      vertical-align: middle;
      white-space: nowrap;
      width: auto;
    }
  ]
})
export class AppComponent implements OnInit {
  public items?: object[];
  public ngOnInit(): void {
    new DataManager({ url: SERVICE_URI, adaptor: new ODataAdaptor() })
      .executeQuery(new Query().group('CustomerID').take(8))
      .then((e: ReturnOption) => this.items = e.result as
object[]).catch((e) => true);
  }
}

```

APP.TEMPLATE.HTML

```

<table class='e-table' >

  <tr>
    <th>Order ID</th>
    <th>Customer ID</th>
    <th>Employee ID</th>
  </tr>
  <tbody group *ngFor="let item of items" [data]='item' ></tbody>
</table>

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Multiple grouping can be done by simply chaining the [group](#) method.

Paging

You can query paged data using [page](#) method. This allow you to query particular set of records based on the page size and index.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ODataAdaptor, ReturnOption } from
 '@syncfusion/ej2-data';
const SERVICE_URI =
 'https://services.odata.org/V3/Northwind/Northwind.svc/Orders/';
@Component({
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',
  styles: [
    .e-table {
      border: solid 1px #e0e0e0;
      border-collapse: collapse;
      font-family: Roboto;
    }
    .e-table td, .e-table th {
      border-style: solid;
      border-width: 1px 0 0;
      border-color: #e0e0e0;
      display: table-cell;
      font-size: 14px;
      line-height: 20px;
      overflow: hidden;
      padding: 8px 21px;
      vertical-align: middle;
      white-space: nowrap;
      width: auto;
    }
  ]
})
export class AppComponent implements OnInit {
  public items?: object[] | any;
  public ngOnInit(): void {
    new DataManager({ url: SERVICE_URI, adaptor: new ODataAdaptor()})
      .executeQuery(new Query().page(2, 6))
      .then((e: ReturnOption) => this.items = e.result as
object[]).catch((e) => true);
  }
}
```

APP.TEMPLATE.HTML

```
<table class='e-table'>
  <tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
  <tr *ngFor="let item of items">

    <td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
/td>
```

```
</tr>
</table>
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Aggregation

The [aggregate](#) method allows you to get aggregated value for a field based on the type.

The built-in aggregate types are,

- sum
- average
- min
- max
- count
- truecount
- falsecount

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ODataAdaptor, ReturnOption } from
 '@syncfusion/ej2-data';
const SERVICE_URI =
 'https://services.odata.org/V3/Northwind/Northwind.svc/Orders/';
@Component({
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',
  styles: [
    .e-table {
      border: solid 1px #e0e0e0;
      border-collapse: collapse;
      font-family: Roboto;
    }
    .e-table td, .e-table th {
      border-style: solid;
      border-width: 1px 0 0;
      border-color: #e0e0e0;
      display: table-cell;
      font-size: 14px;
      line-height: 20px;
      overflow: hidden;
      padding: 8px 21px;
      vertical-align: middle;
      white-space: nowrap;
      width: auto;
```

```

    }
  `]
})
export class AppComponent implements OnInit {
  public items?: object[] | any;
  public min = 0;
  public ngOnInit(): void {
    new DataManager({ url: SERVICE_URI, adaptor: new ODataAdaptor() })
      .executeQuery(new Query().take(5).requiresCount().aggregate('min',
        'EmployeeID'))
      .then((e: ReturnOption) => { this.items = e.result as object[];
    this.min = (e as any).aggregates['EmployeeID - min']; }).catch((e) => true);
  }
}

```

APP.TEMPLATE.HTML

```

<table class='e-table'>
  <tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
  <tr *ngFor="let item of items">

<td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
/td>
  </tr>
  <tr><td></td><td></td><td>Min: {{min}}</td></tr>
</table>

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Hierarchical query

You can use the [hierarchy](#) method to build nested query.

The hierarchical queries are commonly required when you use foreign key binding.

The [foreignKey](#) method is used to specify the key field of the

foreign table and the second argument of the [hierarchy](#) method

accepts a selector function which selects the records from the foreign table.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ODataAdaptor, ReturnOption } from
'@syncfusion/ej2-data';
const SERVICE_URI =
'https://services.odata.org/V3/Northwind/Northwind.svc/Orders/';
@Component({
  standalone: true,

```



```

selector: 'app-root',
templateUrl: './app.template.html',
styles: [
    .e-table {
        border: solid 1px #e0e0e0;
        border-collapse: collapse;
        font-family: Roboto;
    }
    .e-table td, .e-table th {
        border-style: solid;
        border-width: 1px 0 0;
        border-color: #e0e0e0;
        display: table-cell;
        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
    }
]
})
export class AppComponent implements OnInit {
    public items?: object[] | any;
    public ngOnInit(): void {
        new DataManager({ url: SERVICE_URI, adaptor: new ODataAdaptor() })
            .executeQuery(new Query().from('Orders').take(3).hierarchy(
                new Query()
                    .foreignKey('OrderID')
                    .from('Order_Details')
                    .sortBy('Quantity'),
                () => [10248, 10249, 10250] // Selective loading of child
            elements
            ))
            .then((e: ReturnOption) => this.items = e.result as
object[]).catch((e) => true);
    }
}

```

APP.TEMPLATE.HTML

```

<table class='e-table'>
    <tr>
        <th>Order ID</th>
        <th>Customer ID</th>
        <th>Employee ID</th>
    </tr>
    <tbody *ngFor="let item of items">
        <tr>
            <td>{{item.OrderID}}</td>
            <td>{{item.CustomerID}}</td>
            <td>{{item.EmployeeID}}</td>
        </tr>
        <tr>
            <td colspan="3">

```

```

        <table class='e-table'>
            <tr>
                <th>ID</th>
                <th>Price</th>
                <th>Quantity</th>
            </tr>
            <tbody *ngFor="let order of item.Order_Details">
                <tr>
                    <td>{{order.ProductID}}</td>
                    <td>{{order.UnitPrice}}</td>
                    <td>{{order.Quantity}}</td>
                </tr>
            </tbody>
        </table>
    </td>
</tr>
</tbody>
</table>

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Manipulation in Angular Data component

In this section, you will see in detail about how to manipulate data using [DataManager](#). The [DataManager](#) can create, update and delete records either in local data source or remote data source.

Each data sources uses different way in handling the CRUD operations and hence [DataManager](#) uses data adaptors to manipulate data that can be understood by a particular data source.

Insert

The [insert](#) method of [DataManager](#) is used to add new record to the data source. For remote data source, the new record will be send along with the request to the server.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { CommonModule } from '@angular/common';
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ReturnOption } from '@syncfusion/ej2-data';
import { data } from './datasource';
@Component({
  imports: [

    FormsModule , CommonModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',
  styles: [ `

```

```

        .e-table {
            border: solid 1px #e0e0e0;
            border-collapse: collapse;
            font-family: Roboto;
        }
        .e-table td, .e-table th {
            border-style: solid;
            border-width: 1px 0 0;
            border-color: #e0e0e0;
            display: table-cell;
            font-size: 14px;
            line-height: 20px;
            overflow: hidden;
            padding: 8px 21px;
            vertical-align: middle;
            white-space: nowrap;
            width: auto;
        }
        .e-form {
            display: block;
            padding-bottom: 10px;
        }
        .e-form input {
            width: 15%;
        }
    `]
})
export class AppComponent implements OnInit {
    public items?: object[] | any;
    public edit?: { OrderID: string, CustomerID: string, EmployeeID: string }
    | any;
    public dm?: DataManager;
    public text?: string;
    public show = true;
    public ngOnInit(): void {
        this.text = 'Insert';
        this.edit = { OrderID: null, CustomerID: null, EmployeeID: null } as
any;
        this.dm = new DataManager(data.slice(0, 5));
        this.dm.executeQuery(new Query())
            .then((e: ReturnOption) => this.items = e.result as
object[]).catch((e) => true);
    }
    public insert(): void {
        (this.dm as DataManager).insert({
            OrderID: this.edit?.OrderID,
            CustomerID: this.edit?.CustomerID,
            EmployeeID: this.edit?.EmployeeID
        });
        (this.dm as DataManager).executeQuery(new Query())
            .then((e: ReturnOption) => this.items = e.result as
object[]).catch((e) => true);
    }
}

```

APP.TEMPLATE.HTML

```

<div class="e-form">
  <input type="number" id='OrderID' placeholder="Order ID"
  [(ngModel)]="edit.OrderID" />
  <input type="text" id="CustomerID" placeholder="Customer ID"
  [(ngModel)]="edit.CustomerID" *ngIf="show"/>
  <input type="number" id="EmployeeID" placeholder="Employee ID"
  [(ngModel)]="edit.EmployeeID" *ngIf="show"/>
  <input type="button" [value]="text" id="manipulate" (click)="insert()" />
</div>
<table class='e-table'>
  <tr>
    <th>Order ID</th>
    <th>Customer ID</th>
    <th>Employee ID</th>
  </tr>
  <tr *ngFor="let item of items">
    <td>{{item.OrderID}}</td>
    <td>{{item.CustomerID}}</td>
    <td>{{item.EmployeeID}}</td>
  </tr>
</table>

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

In remote data sources, when the primary key field is an identity field, then it is advised to return the created data in the response.

Update

The [update](#) method of

`DataManager` (<https://ej2.syncfusion.com/documentation/api/data/dataManager/>) is used to modify/update a record in the data source. For remote data source, the modified record will be send along with the request to the server.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { CommonModule } from '@angular/common';
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ReturnOption } from '@syncfusion/ej2-data';
import { data } from './datasource';
@Component({
  imports: [

    FormsModule, CommonModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',

```

```

        styles: [
            {
                '.e-table {
                    border: solid 1px #e0e0e0;
                    border-collapse: collapse;
                    font-family: Roboto;
                }
                .e-table td, .e-table th {
                    border-style: solid;
                    border-width: 1px 0 0;
                    border-color: #e0e0e0;
                    display: table-cell;
                    font-size: 14px;
                    line-height: 20px;
                    overflow: hidden;
                    padding: 8px 21px;
                    vertical-align: middle;
                    white-space: nowrap;
                    width: auto;
                }
                .e-form {
                    display: block;
                    padding-bottom: 10px;
                }
                .e-form input {
                    width: 15%;
                }
            }
        ]
    })
    export class AppComponent implements OnInit {
        public items?: object[] | any;
        public edit?: { OrderID: string, CustomerID: string, EmployeeID: string }
        | any;
        public dm?: DataManager;
        public text?: string;
        public show = true;
        public ngOnInit(): void {
            this.text = 'Update';
            this.edit = { OrderID: null, CustomerID: null, EmployeeID: null } as
any;
            this.dm = new DataManager(data.slice(0, 5));
            this.dm.executeQuery(new Query())
                .then((e: ReturnOption) => this.items = e.result as
object[]).catch((e) => true);
        }
        public insert(): void {
            (this.dm as DataManager).update('OrderID', {
                OrderID: this.edit?.OrderID,
                CustomerID: this.edit?.CustomerID,
                EmployeeID: this.edit?.EmployeeID
            });
            (this.dm as DataManager).executeQuery(new Query())
                .then((e: ReturnOption) => this.items = e.result as
object[]).catch((e) => true);
        }
    }
}

```

APP.TEMPLATE.HTML

```

<div class="e-form">
  <input type="number" id='OrderID' placeholder="Order ID"
  [(ngModel)]="edit.OrderID" />
  <input type="text" id="CustomerID" placeholder="Customer ID"
  [(ngModel)]="edit.CustomerID" *ngIf="show"/>
  <input type="number" id="EmployeeID" placeholder="Employee ID"
  [(ngModel)]="edit.EmployeeID" *ngIf="show"/>
  <input type="button" [value]="text" id="manipulate" (click)="insert()" />
</div>
<table class='e-table'>
  <tr>
    <th>Order ID</th>
    <th>Customer ID</th>
    <th>Employee ID</th>
  </tr>
  <tr *ngFor="let item of items">
    <td>{{item.OrderID}}</td>
    <td>{{item.CustomerID}}</td>
    <td>{{item.EmployeeID}}</td>
  </tr>
</table>

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Primary key name is required by the [update](#) method to find the record to be updated.

Remove

The [remove](#) method of [DataManager](#) is used to remove a record from the data source. For remote data source, the record details such as primary key and data will be send along with the request to the server.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { CommonModule } from '@angular/common';
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ReturnOption } from '@syncfusion/ej2-data';
import { data } from './datasource';
@Component({
  imports: [

    FormsModule , CommonModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',
  styles: [ `
    .e-table {

```

```

        border: solid 1px #e0e0e0;
        border-collapse: collapse;
        font-family: Roboto;
    }
    .e-table td, .e-table th {
        border-style: solid;
        border-width: 1px 0 0;
        border-color: #e0e0e0;
        display: table-cell;
        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
    }
    .e-form {
        display: block;
        padding-bottom: 10px;
    }
    .e-form input {
        width: 15%;
    }
    `]
  })
  export class AppComponent implements OnInit {
    public items?: object[] | any;
    public edit?: { OrderID: string, CustomerID: string, EmployeeID: string }
    | any;
    public dm?: DataManager;
    public text?: string;
    public show = false;
    public ngOnInit(): void {
      this.text = 'Remove';
      this.edit = { OrderID: null, CustomerID: null, EmployeeID: null } as
any;
      this.dm = new DataManager(data.slice(0, 5));
      this.dm.executeQuery(new Query())
        .then((e: ReturnOption) => this.items = e.result as
object[]).catch((e) => true);
    }
    public insert(): void {
      (this.dm as DataManager).remove( 'OrderID', {
        OrderID: this.edit.OrderID
      });
      (this.dm as DataManager).executeQuery(new Query())
        .then((e: ReturnOption) => this.items = e.result as
object[]).catch((e) => true);
    }
  }
}

```

APP.TEMPLATE.HTML

```
<div class="e-form">
```

```

    <input type="number" id='OrderID' placeholder="Order ID"
    [(ngModel)]="edit.OrderID" />
    <input type="text" id="CustomerID" placeholder="Customer ID"
    [(ngModel)]="edit.CustomerID" *ngIf="show"/>
    <input type="number" id="EmployeeID" placeholder="Employee ID"
    [(ngModel)]="edit.EmployeeID" *ngIf="show"/>
    <input type="button" [value]="text" id="manipulate" (click)="insert()" />
</div>
<table class='e-table'>
  <tr>
    <th>Order ID</th>
    <th>Customer ID</th>
    <th>Employee ID</th>
  </tr>
  <tr *ngFor="let item of items">
    <td>{{item.OrderID}}</td>
    <td>{{item.CustomerID}}</td>
    <td>{{item.EmployeeID}}</td>
  </tr>
</table>

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Primary key name and its value are required to find the record to be removed.

Batch Edit Operation

[DataManager](#) supports batch processing for the CRUD operations. You can use the [saveChanges](#) method to batch the edit operation. For remote data source, requests to add, remove and change are handled altogether at a time rather than passing the request separately for each operation.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { CommonModule } from '@angular/common';
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ReturnOption } from '@syncfusion/ej2-data';
import { data } from './datasource';
@Component({
  imports: [

    FormsModule, CommonModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',
  styles: [`
    .e-table {
      border: solid 1px #e0e0e0;

```



```

        border-collapse: collapse;
        font-family: Roboto;
    }
    .e-table td, .e-table th {
        border-style: solid;
        border-width: 1px 0 0;
        border-color: #e0e0e0;
        display: table-cell;
        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
    }
    .e-form {
        display: block;
        padding-bottom: 10px;
    }
    .e-form input {
        width: 15%;
    }
    `]
    })
    export class AppComponent implements OnInit {
        public items?: object[] | any;
        public edit?: { OrderID: string, CustomerID: string, EmployeeID: string }
    | any;
        public dm?: DataManager;
        public changes: { changedRecords: object[], addedRecords: object[],
        deletedRecords: object[] } =
        { changedRecords: [], addedRecords: [], deletedRecords: [] };
        public text?: string;
        public ngOnInit(): void {
            this.text = 'Update';
            this.edit = { OrderID: null, CustomerID: null, EmployeeID: null } as
any;
            this.dm = new DataManager(data.slice(0, 5));
            this.dm.executeQuery(new Query())
                .then((e: ReturnOption) => this.items = e.result as
object[]).catch((e) => true);
        }
        public save(): void {
            (this.dm as DataManager).saveChanges(this.changes);
            (this.dm as DataManager).executeQuery(new Query())
                .then((e: ReturnOption) => this.items = e.result as
object[]).catch((e) => true);
            this.changes = { changedRecords: [], addedRecords: [],
        deletedRecords: [] };
        }
        public insert(action: string): void {
            (this.changes as any)[action].push({
                OrderID: this.edit?.OrderID,
                CustomerID: this.edit?.CustomerID,
                EmployeeID: this.edit?.EmployeeID
            });
        }
    }

```

```

        this.edit = { OrderID: null, CustomerID: null, EmployeeID: null } as
any;
    }
}

```

APP.TEMPLATE.HTML

```

<div class="e-form">
    <input type="number" id='OrderID' placeholder="Order ID"
[(ngModel)]="edit.OrderID" />
    <input type="text" id="CustomerID" placeholder="Customer ID"
[(ngModel)]="edit.CustomerID"/>
    <input type="number" id="EmployeeID" placeholder="Employee ID"
[(ngModel)]="edit.EmployeeID"/>
    <input type="button" value="Insert" (click)="insert('addedRecords')" />
    <input type="button" value="Update" (click)="insert('changedRecords')" />
    <input type="button" value="Remove" (click)="insert('deletedRecords')" />
</div>
<div class="e-form">
    <label>Click to Save changes: </label>
    <input type="button" value="Save Changes" (click)="save()" />
</div>
<table class='e-table'>
    <tr>
        <th>Order ID</th>
        <th>Customer ID</th>
        <th>Employee ID</th>
    </tr>
    <tr *ngFor="let item of items">
        <td>{{item.OrderID}}</td>
        <td>{{item.CustomerID}}</td>
        <td>{{item.EmployeeID}}</td>
    </tr>
</table>

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

How to in Angular Data component

Work in offline mode

On remote data binding, every time invoking [executeQuery](#) will send request to the server and the query will be processed on server-side.

To avoid post back to server on calling [executeQuery](#), you can set the [DataManager](#) to load all the data on initialization time and make the query processing in client-side.

To enable this behavior, you can use **offline** property of [DataManager](#).

`typescript

```
import { Component, OnInit } from '@angular/core';
```

```
import { DataManager, Query, ODataAdaptor, ReturnOption } from '@syncfusion/ej2-data';
const SERVICE_URI = 'https://services.odata.org/V3/Northwind/Northwind.svc/Orders/';
@Component({
  selector: 'app-root',
  templateUrl: 'app.template.html',
  styles: [`
    .e-table {
      border: solid 1px #e0e0e0;
      border-collapse: collapse;
      font-family: Roboto;
    }
    .e-table td, .e-table th {
      border-style: solid;
      border-width: 1px 0 0;
      border-color: #e0e0e0;
      display: table-cell;
      font-size: 14px;
      line-height: 20px;
      overflow: hidden;
      padding: 8px 21px;
      vertical-align: middle;
      white-space: nowrap;
      width: auto;
    }
  `]
})
export class AppComponent implements OnInit {
  public items: object[];
  public ngOnInit(): void {
    const dm: DataManager = new DataManager(
      { url: SERVICE_URI, adaptor: new ODataAdaptor(), offline: true },
      new Query().take(8)
    );
```

```

dm.ready.then((e: ReturnOption) => this.items = e.result as object[]).catch((e) => true);
}
}
,

```

The loaded data will be cached in the **json** property of [DataManager](#).

Sending additional parameters to server

You can use the [addParams](#) method of [Query](#) class, to add custom parameter to the data request.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { CommonModule } from '@angular/common';
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, ODataAdaptor, ReturnOption } from
 '@syncfusion/ej2-data';
const SERVICE_URI =
 'https://services.odata.org/V3/Northwind/Northwind.svc/Orders/';
@Component({
  imports: [CommonModule],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.template.html',
  styles: [
    `
      .e-table {
        border: solid 1px #e0e0e0;
        border-collapse: collapse;
        font-family: Roboto;
      }
      .e-table td, .e-table th {
        border-style: solid;
        border-width: 1px 0 0;
        border-color: #e0e0e0;
        display: table-cell;
        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
      }
    `
  ]
})
export class AppComponent implements OnInit {
  public items?: object[] | any;
  public ngOnInit(): void {
    new DataManager({ url: SERVICE_URI, adaptor: new ODataAdaptor() })
      .executeQuery(new Query().addParams('$top', '8')).then((e:
ReturnOption) => this.items = e.result as object[]).catch((e) => true);
  }
}

```

APP.TEMPLATE.HTML

```
<table class='e-table'>
  <tr><th>Order ID</th><th>Customer ID</th><th>Employee ID</th></tr>
  <tr *ngFor="let item of items">

<td>{{item.OrderID}}</td><td>{{item.CustomerID}}</td><td>{{item.EmployeeID}}<
/td>
  </tr>
</table>
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Adding custom headers

You can add custom headers to the request made by [DataManager](#) using the **headers** property.

```
`typescript
```

```
import { DataManager, Query, ReturnOption, ODataAdaptor } from '@syncfusion/ej2-data';
const SERVICE_URI = 'https://services.odata.org/V3/Northwind/Northwind.svc/Orders/';
new DataManager({ url: SERVICE_URI, adaptor: new ODataAdaptor, headers: [{ 'syncfusion': 'true' }] })
.executeQuery(new Query())
.then((e: ReturnOption) => {
//get result from e.result
});
```

[Link to the Video`](#)

Adding custom headers while making cross domain request will initiate preflight request.

DatePicker

Getting started with Angular Datepicker component

The following section explains the steps required to create a simple DatePicker component and also it demonstrates the basic usage of the DatePicker.

To get start quickly with Angular DatePicker component, refer to the video below.

Dependencies

Install the below required dependency package in order to use the **DatePicker** component in your application.

```
`javascript
```

```
|-- @syncfusion/ej2-angular-calendars
```

```
|-- @syncfusion/ej2-angular-base
```

```
|-- @syncfusion/ej2-base  
|-- @syncfusion/ej2-calendars  
|-- @syncfusion/ej2-lists  
|-- @syncfusion/ej2-inputs  
|-- @syncfusion/ej2-splitbuttons  
|-- @syncfusion/ej2-popups  
|-- @syncfusion/ej2-buttons  
,
```

Setup Angular environment

Angular provides the easiest way to set angular CLI projects using [Angular CLI](#) tool.

Install the CLI application globally to your machine.

```
`bash  
npm install -g @angular/cli  
,
```

Create a new application

```
`bash  
ng new syncfusion-angular-datepicker  
,
```

By default, it install the CSS style base application. To setup with SCSS, pass `--style=scss` argument on create project.

Example code snippet.

```
`bash  
ng new syncfusion-angular-datepicker --style=scss  
,
```

Navigate to the created project folder.

```
`bash  
cd syncfusion-angular-datepicker  
,
```

Installing Syncfusion DatePicker package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(>=20.2.36) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-calendars](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-calendars --save
`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the [ngcc](#) package use the below.

Add [@syncfusion/ej2-angular-calendars@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-calendars@ngcc --save
`
```

To mention the ngcc package in the [package.json](#) file, add the suffix [-ngcc](#) with the package version as below.

```
`bash
@syncfusion/ej2-angular-calendars:"20.2.38-ngcc"
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Registering DatePicker module

Import DatePicker module into Angular application(src/app/app.module.ts) from the package [@syncfusion/ej2-angular-calendars](#).

```
`javascript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// import the DatePickerModule for the DatePicker component
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars';
import { AppComponent } from './app.component';
@NgModule({
//declaration of DatePickerModule into NgModule
imports: [ BrowserModule, DatePickerModule ],
declarations: [ AppComponent ],
bootstrap: [ AppComponent ]
})
```

```
})  
export class AppModule { }  
`
```

Adding CSS reference

The following CSS files are available in `../node_modules/@syncfusion` package folder.

This can be referenced in `[src/styles.css]` using following code.

```
`css  
  
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';  
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';  
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';  
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';  
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';  
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';  
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';  
@import '../node_modules/@syncfusion/ej2-angular-calendars/styles/material.css';  
`
```

If you want to refer the combined component styles, please make use of our [CRG](#) (Custom Resource Generator) in your application.

Adding DatePicker component

Modify the template in `[src/app/app.component.ts]` file to render the DatePicker component. by using `<ejs-datepicker>` selector.

```
`javascript  
  
import { Component } from '@angular/core';  
@Component({  
  selector: 'app-root',  
  template: `<!-- To Render DatePicker -->  
  <ejs-datepicker></ejs-datepicker>`  
})  
export class AppComponent { }  
`
```

Running the application

After completing the configuration required to render a basic DatePicker, run the following command to display the output in your default browser.

```
ng serve
```


The following example illustrates the output in your browser

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-datepicker></ejs-datepicker>`
})
export class AppComponent {
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Setting the selected date

To set the selected date, use the [value]

(<https://ej2.syncfusion.com/angular/documentation/api/datepicker#value>) property.

The below example demonstrates the DatePicker with current date as selected one.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-datepicker [value]='dateValue' placeholder='Enter date'></ejs-datepicker>`
})
export class AppComponent {
  public dateValue: Date = new Date();
  constructor() {
  }
}
```

```
}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Setting the date range to restrict selection

To restrict the selection of date within a specified range, use the [min](#) and [max](#) properties. To know more about range restriction in DatePicker, please refer this [page](#).

The below example demonstrates the DatePicker to select a date within a range from 7 to 27.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-datepicker id="datepicker" [min]='minDate'
    [max]='maxDate'></ejs-datepicker>
  `
})
export class AppComponent {
  public month: number = new Date().getMonth();
  public fullYear: number = new Date().getFullYear();
  public minDate: Date = new Date(this.fullYear, this.month, 7);
  public maxDate: Date = new Date(this.fullYear, this.month, 27);
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Change the format of selected date](#)
- [Render DatePicker with specific culture](#)

- [How to change the initial view of the DatePicker](#)
- [How to achieve validation with DatePicker](#)
- [How to achieve two-way binding with DatePicker](#)

Date range in Angular Datepicker component

You can restrict the user to select the date from the specified range of dates by using the [min](#) and [max](#) properties.

When the min and max properties are configured and the selected date value is out-of-range or invalid, then the model value will be set to **out of range** date value or **null** respectively with highlighted **error** class to indicates the date is out of range or invalid.

The below example demonstrates the Calendar to select a date within a range from 1 to 27 in a month.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-datepicker [value]='dateValue' [min]='minDate'
    [max]='maxDate'></ejs-datepicker>
  `
})
export class AppComponent {
  public today: Date = new Date();
  public currentYear: number = this.today.getFullYear();
  public currentMonth: number = this.today.getMonth();
  public currentDay: number = this.today.getDate();
  public dateValue: Object = new Date(new Date().setDate(14));
  public minDate: Object = new Date(this.currentYear, this.currentMonth,
1);
  public maxDate: Object = new Date(this.currentYear, this.currentMonth,
27);
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Globalization in Angular Datepicker component

Globalization is the combination of internalization and localization. You can adapt the component to various languages by parsing and formatting the date or number [Internationalization](#) and also add culture specific customization and translation to the text [localization](#).

By default, DatePicker date format, week and month names are specific to English culture. It utilizes the [Essential JavaScript 2 Internationalization](#) package to parse and format the date object based on the culture by using the official [UNICODE CLDR](#)

JSON data. It provides the `loadCldr` method to load culture specific CLDR JSON data. To use a different culture other than English, follow the steps below:

- Install the `CLDR-Data` package by using the following command (installs all the CLDR JSON data). To know more about CLDR-Data refer to the [CLDR-Data](#) link.

,

```
npm install cldr-data --save
```

,

Once the package installed, you can find the culture specific JSON data under the location `/node_modules/cldr-data`.

- Now import the installed CLDR JSON data into the `app.component.ts` file.
- Now use the `loadCldr` method to load the culture specific CLDR JSON data from the installed location to `app.component.ts` file.
- DatePicker displayed Sunday as the first day of week based on default culture ("en-US"). If you want to display the DatePicker with loaded culture's first day of week, you need to import `weekdata.json` file from the `cldr-data/supplemental` as given in the code example.

```
`typescript
```

```
import { loadCldr } from "@syncfusion/ej2-base";
```

```
declare var require: any;
```

```
loadCldr(
```

```
  require("cldr-data/main/de/numbers.json"),
```

```
  require("cldr-data/main/de/ca-gregorian.json"),
```

```
  require("cldr-data/supplemental/numberingSystems.json"),
```

```
  require("cldr-data/main/de/timeZoneNames.json"),
```

```
  require('cldr-data/supplemental/weekdata.json') // To load the culture based first day of week
```

```
);
```

,

The **Localization** library allows you to localize default text content of the DatePicker. The DatePicker component has static text for **today** feature that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

Locale keywords | Text

today | Name of the button to choose Today date.

placeholder | Hint to describe expected value in input element.

- Before changing to a culture other than **English**, ensure that locale text for the concerned culture is loaded through **load** method of **L10n** class.

`typescript

```
//Load the L10n, loadCldr from ej2-base
import { loadCldr, L10n } from "@syncfusion/ej2-base";
//load the locale object to set the localized placeholder value
L10n.load({
  de: {
    datepicker: {
      placeholder: "Wählen Sie ein Datum",
      today:"heute"
    }
  }
});
`
```

- Set the culture by using the **locale** property.

In the following code example, the DatePicker is initialized in **German** culture with corresponding localized text.

The following example demonstrates the DatePicker in **German** culture.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
import { DatePickerComponent } from '@syncfusion/ej2-angular-calendars';
import { loadCldr, L10n } from '@syncfusion/ej2-base';
// Here we have referred local json files for preview purpose
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as detimeZoneNames from './timeZoneNames.json';
```

```
loadCldr(numberingSystems, gregorian, numbers, detimeZoneNames);
@Component({
  imports: [
    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-datepicker locale='de'></ejs-datepicker>`
})
export class AppComponent {
  ngOnInit(): void {
    /*loads the localization text*/
    L10n.load({
      'de': {
        'datepicker': {
          placeholder: 'Wählen Sie ein Datum aus',
          today: 'heute'
        }
      }
    });
  }
  constructor() {}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Right-To-Left

The DatePicker supports RTL (right-to-left) functionality for languages like Arabic and Hebrew to displays the text in the right-to-left direction. Use `enableRtl` property to set the RTL direction.

The following code example initialize the DatePicker component in Arabic culture and also explains how to set the localized text to the placeholder using `load` method of `L10n` class.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
import { DatePickerComponent } from '@syncfusion/ej2-angular-calendars';
import { loadCldr, L10n } from '@syncfusion/ej2-base';
// Here we have referred local json files for preview purpose
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as artimeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, artimeZoneNames);
@Component({
  imports: [
```

```

        DatePickerModule
    ],
    standalone: true,
    selector: 'app-root',
    template: `<ejs-datepicker enableRtl='true' locale='ar'></ejs-
datepicker>`
  })
  export class AppComponent {
    ngOnInit(): void {
      L10n.load({
        'ar': {
          'datepicker': {
            placeholder: "اختر تاريخا",
            today: "اليوم"
          }
        }
      });
    }
    constructor() {}
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Date format in Angular Datepicker component

Date Format

Date format is a way of representing the date value in different string format in textbox.

By default the DatePicker's format is based on the culture. You can also set the own custom format by using the [format](#) property.

Once the date format property has been defined it will be common to all the cultures.

To know more about the date format standards, refer to the [Internationalization Date Format](#) section.

The following example demonstrates the DatePicker with the custom format (yyyy-MM-dd).

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [

    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-datepicker format='yyyy-MM-dd' placeholder='Enter date'

```

```

    [value]=dateValue></ejs-datepicker>`
  })
  export class AppComponent {
    public dateValue: Date = new Date("05/17/2017");
    constructor() {
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Parse and Format Date value based on culture-specific formats

Parse date is a process of converting string value into a date object by using the [parseDate](#) method. This method takes two arguments, the string value and [DateFormatOptions](#). Then, returns the date Object.

Format date is a process of converting date object into a formatted string value by using the [formatDate](#) method. This method takes two arguments, the date object and [DateFormatOptions](#). Then, returns the formatted string.

The following example demonstrates how to parse the date value and format the date value based on the **German** culture and **dd MMMM yyyy** date format. For every value change, the changed date object value will be formatted into a string and the text value of the component will be parsed into a date object. These values are showcased in the example.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from "@angular/core";
import { DatePickerComponent } from "@syncfusion/ej2-angular-calendars";
import { loadCldr, L10n } from '@syncfusion/ej2-base';
// Here we have referred local json files for preview purpose
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as detimeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, detimeZoneNames);
@Component({
  imports: [

    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-datepicker id="datepicker" #dateObj locale="de" width="245px"
    (change)="onChange($event) "format="dd MMMM yyyy"></ejs-datepicker>
    <div class="valuestring">
      <span>Parsed value</span>: <br />
      <span id="parsed"></span>
    `
})

```



```

    <br /><br />
    <span>Formatted value </span>: <br />
    <span id="formatted"></span>
  </div>`
})
export class AppComponent {
  @ViewChild('dateObj')
  public dateObj?: any;
  ngOnInit(): void {
    /*loads the localization text*/
    L10n.load({
      'de': {
        'datepicker': {
          placeholder: 'Wählen Sie ein Datum aus',
          today: 'heute'
        }
      }
    });
  }
  constructor() {}
  onChange(args: any) {
    if (args.value) {
      (document.getElementById('parsed') as HTMLElement).innerText =
this.dateObj.globalize.parseDate(this.dateObj.inputElement.value, {
        format: 'dd MMMM yyyy',
        type: 'dateTime',
        skeleton: 'yMd',
      });
      (document.getElementById('formatted') as HTMLElement).innerText =
this.dateObj.globalize.formatDate(this.dateObj.value, {
        format: 'dd MMMM yyyy',
        type: 'dateTime',
        skeleton: 'yMd',
      });
    } else {
      (document.getElementById('parsed') as HTMLElement).innerText = '';
      (document.getElementById('formatted') as HTMLElement).innerText = '';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Date masking in Angular Datpicker component

The DatePicker has built-in support to masking the date value, when `enableMask` property set as `true`.

To use mask support, inject the MaskedDateTime module in the DatePicker.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';

```

```
import { BrowserModule } from '@angular/platform-browser';
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars';
import { Component } from '@angular/core';
import { MaskedDateTimeService } from '@syncfusion/ej2-angular-calendars';
@Component({
  imports: [
    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-datepicker [enableMask]="enableMaskSupport"></ejs-datepicker>
  `,
  providers: [MaskedDateTimeService],
})
export class AppComponent {
  constructor() {
    }
    public enableMaskSupport: boolean = true;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

The mask pattern is defined based on the provided date format to the component. If the format is not specified, the mask pattern is formed based on the default format of the current culture.

The selected portions of date and time co-ordinates can be incremented and decremented using the Up/Down arrow keys. You can also use Right/Left arrow keys to navigate from one segment to another.

The following example demonstrates default and custom format of DatePicker component with mask.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars';
import { Component } from '@angular/core';
import { MaskedDateTimeService } from '@syncfusion/ej2-angular-calendars';
@Component({
  imports: [
    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './format.html',
  providers: [MaskedDateTimeService],
})
export class AppComponent {
  constructor() {
    }
    public format: string = "dd/MM/yyyy";
  }
}
```

```
public enableMaskSupport: boolean = true;
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Configure Mask Placeholder

You can change mask placeholder value through property `maskPlaceholder`. By default, it takes the full name of date and time co-ordinates such as `day`, `month`, `year`, `hour` etc.

While changing to a culture other than `English`, ensure that locale text for the concerned culture is loaded through load method of `L10n` class for mask placeholder values like below.

`typescript

//Load the L10n from ej2-base

```
import { L10n } from '@syncfusion/ej2-base';
```

//load the locale object to set the localized mask placeholder value

```
L10n.load({
  'de': {
    datepicker: { day: 'Tag', month: 'Monat', year: 'Jahr' }
  }
});
`
```

The following example demonstrates default and customized mask placeholder value.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars';
import { Component } from '@angular/core';
import { MaskedDateTimeService } from '@syncfusion/ej2-angular-calendars';
@Component({
  imports: [
    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './maskplaceholder.html',
  providers: [MaskedDateTimeService],
})
export class AppComponent {
  constructor() {
  }
}
```

```

    public enableMaskSupport: boolean = true;
    public maskPlaceholderValue: Object = {day: 'd', month: 'M', year: 'Y'}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Strict mode in Angular Datepicker component

The [strictMode](#) is an act, that allows the user to enter only the valid date within the specified min/max range in textbox.

If the date is invalid, then the component will stay with the previous value.

Else, if the date is out of range, then the component will set the date to the min/max date.

The following example demonstrates the DatePicker in `strictMode` with min/max range of 5th to 25th in a month of May.

Here, it allows to enter only the valid date within the specified range. If you are trying to enter the out-of-range value as like 28th of May, then the value will set to the max date of 25th May. Since the value 28th is greater than to max value of 25th.

Or else if you are trying to enter the invalid date, then the value will stay with the previous value.

The following example demonstrates the DatePicker with `strictMode true`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars';
import { Component } from '@angular/core';

@Component({
  imports: [
    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-datepicker strictMode='true' format='dd/MM/yyyy'
placeholder='Enter date' [value]='dateValue' [min]='minDate'
[max]='maxDate'></ejs-datepicker>`
})
export class AppComponent {
  public dateValue: Date = new Date("5/28/2018");
  public minDate : Date = new Date ("5/5/2018");
  public maxDate : Date = new Date ("5/25/2018");
  constructor() {
  }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

By default, the DatePicker act in strictMode **false** state, that allows to enter the invalid or out-of-range date in textbox.

If the date is out-of-range or invalid, then the model value will be set to **out of range** date value or **null** respectively with highlighted **error** class to indicates the date is out of range or invalid.

The following example demonstrates the **strictMode** as **false**. Here, it allows to enter the valid or invalid value in textbox.

If you are entering out-of-range or invalid date value, then the model value will be set to **out of range** date value or **null** respectively with highlighted **error** class to indicates the date is out of range or invalid.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-datepicker format='dd/MM/yyyy' placeholder='Enter date'
[value]='dateValue' [min]='minDate' [max]='maxDate'></ejs-datepicker>`
})
export class AppComponent {
  public dateValue: Date = new Date("5/28/2018");
  public minDate : Date = new Date ("5/5/2018");
  public maxDate : Date = new Date ("5/25/2018");
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

If the value of min or max properties changed through code behind, then you have to update the **value** property to set within the range.

Customization in Angular Datepicker component

You can customize the entire appearance of the input element and Calendar by using custom [cssClass](#) property.

Also you can use the calendar's [renderDayCell](#) event to customize the appearance of the each day cell.

Below is the list of classes that provides flexible way to customize the DatePicker component.

Class Name	Description
---	---
e-date-wrapper	Applied to DatePicker wrapper
e-datepicker	Applied to the DatePicker element.
e-float-text	Applied to the floating label.
e-date-icon	Applied to the DatePicker icon.
e-popup-wrapper	Applied to DatePicker popup wrapper.
e-calendar	Applied to Calendar element.
e-header	Applied to Calendar header.
e-title	Applied to Calendar title.
e-icon-container	Applied to Calendar previous and next icon container.
e-prev	Applied to Calendar previous icon.
e-next	Applied to Calendar next icon.
e-weekend	Applied to Calendar weekend dates.
e-other-month	Applied to Calendar other month dates.
e-day	Applied to each day cell of the Calendar.
e-selected	Applied to Calendar selected dates.
e-disabled	Applied to Calendar disabled dates.

The following example disables the weekends of every month using `renderDayCell` event.

Here we have used the `e-disabled` class to highlight the disabled date.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
import { RenderDayCellEventArgs } from '@syncfusion/ej2-angular-calendars';
@Component({
  imports: [
    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./index.css'],
```

```

    template: `<ejs-datepicker [value]='dateValue' placeholder='Enter date'
(renderDayCell)='onRenderCell($event)'></ejs-datepicker>`
  })
  export class AppComponent {
    public dateValue:Date = new Date();
    constructor() {
    }
    public onRenderCell(args: RenderDayCellEventArgs): void {
    if (args.date?.getDay() == 0 || args.date?.getDay() == 6) {
      //sets isDisabled to true to disable the date.
      args.isDisabled = true;
      //To know about the disabled date customization, you can refer in
      "styles.css".
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Adding mandatory asterisk to placeholder and float label

You can add a mandatory asterisk(*) to placeholder and float label using `.e-input-group.e-control-wrapper.e-float-input.e-float-text::after` class.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [

    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-datepicker floatLabelType="auto" placeholder="Enter
date"></ejs-datepicker>`
})
export class AppComponent {
  constructor() {
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Render DatePicker with CSS customization](#)
- [How to disable the DatePicker control](#)
- [How to set read-only for DatePicker](#)
- [How to customize the DatePicker day header](#)

Date views in Angular Datepicker component

The DatePicker has the following predefined views that provides a flexible way to navigate back and forth to select the date.

| View | Description |

| --- | --- |

| month(default) | Displays the days in a month |

| year | Displays the months in a year |

| decade | Displays the years in a decade |

Start view

You can use the [start](#) property to define the initial rendering view.

The following example demonstrates how to create a DatePicker with **decade** as initial rendering view.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-datepicker start='Decade' placeholder='Enter date'></ejs-datepicker>`
})
export class AppComponent {
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```


Depth view restriction

Define the [depth](#) property to control the view navigation.

Always the depth view has to be smaller than the start view, otherwise the view restriction will be not restricted.

The following example demonstrates how to create a DatePicker that allows users to select a month.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-datepicker start='Decade' depth='Year' placeholder='Enter date'></ejs-datepicker>`
})
export class AppComponent {
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

To know more about Calendar views refer the Calendar's [Calendar Views](#) section.

Accessibility in Angular Datepicker component

The DatePicker component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the DatePicker component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

```

| Right-To-Left Support |  |
| Color Contrast |  |
| Mobile Device Support |  |
| Keyboard Navigation Support |  |
| Accessibility Checker Validation |  |
| Axe-core Accessibility Validation |  |

<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

```

WAI-ARIA attributes

The Web accessibility defines a way to make web content and web applications more accessible to disabled people. It especially helps the dynamic content change and advanced user interface controls developed with Ajax, HTML, JavaScript, and related technologies.

DatePicker provides built-in compliance with the [WAI-ARIA](#) specifications. WAI-ARIA supports is achieved through the attributes like `aria-expanded`, `aria-disabled`, `aria-activedescendant` applied for the input element.

It helps to provide information about the widget for assistive technology to the disabled person in a screen reader.

- **Aria-expanded** : attributes indicates the state of a collapsible element.
- **Aria-disabled** : attribute indicates the disabled state of this DatePicker component.
- **Aria-activedescendent** : attribute helps in managing the current active child of the DatePicker component.

For more information about accessibility of Calendar refer to the Calendar [Accessibility](#) section.

Keyboard Interaction

You can use the following keys to interact with the DatePicker.

The component implements the keyboard navigation support by following the [WAI-ARIA practices](#).

It supports the below list of shortcut keys.

Input Element

Press	To do this
---	---
Alt + Down Arrow	Opens the popup.
Alt + Up Arrow	Closes the popup.
Esc	closes the popup.

Calendar Navigation

Use the below list of keys to navigate the Calendar after the popup has opened.

Press	To do this
---	---
Upper Arrow	Focus the previous week date.
Down Arrow	Focus the next week date.
Left Arrow	Focus the previous date.
Right Arrow	Focus the next date.
Home	Focus the first date in the month.
End	Focus the last date in the month.
Page Up	Focus the same date in the previous month.
Page Down	Focus the same date in the next month.
Enter	Select the currently focused date.
Shift + Page Up	Focus the same date in the previous year.
Shift + Page Down	Focus the same date in the next year.
Control + Upper Arrow	Moves into the inner level of view like month-year, year-decade
Control + Down Arrow	Moves out from the depth level view like decade-year, year-month
Control + Home	Focus the starting date in the current year.
Control + End	Focus the ending date in the current year.

To focus the DatePicker component use the **alt+t** keys.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
```

```
import { Component, HostListener, ViewChild } from '@angular/core';
import { DatePickerComponent } from '@syncfusion/ej2-angular-calendars';
@Component({
  imports: [
    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-datepicker #ejDatePicker [value]='dateValue' placeholder='Enter
    date'></ejs-datepicker>
  `
})
export class AppComponent {
  @ViewChild('ejDatePicker') ejDatePicker?: DatePickerComponent;
  public dateValue: Date = new Date();
  @HostListener('document:keyup', ['$event'])
  handleKeyboardEvent(event: KeyboardEvent) {
    if (event.altKey && event.keyCode === 84 /* t */) {
      // press alt+t to focus the control.
      this.ejDatePicker?.focusIn();
    }
  }
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Ensuring accessibility

The DatePicker component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the DatePicker component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the DatePicker component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Style appearance in Angular Datepicker component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of DatePicker wrapper element

Use the following CSS to customize the appearance of wrapper element.

```
`css
/ To specify height and font size /
.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input {
height: 40px;
font-size: 20px;
}
`
```

Customizing the DatePicker icon element

Use the following CSS to customize the DatePicker icon element

```
`css
/ To specify background color and font size /
.e-input-group .e-input-group-icon:last-child, .e-input-group.e-control-wrapper .e-input-group-icon:last-child {
font-size: 12px;
background-color: darkgray;
}
`
```

Customizing the Calendar popup of the DatePicker

Please check the below section, to customize the style and appearance of the Calendar component

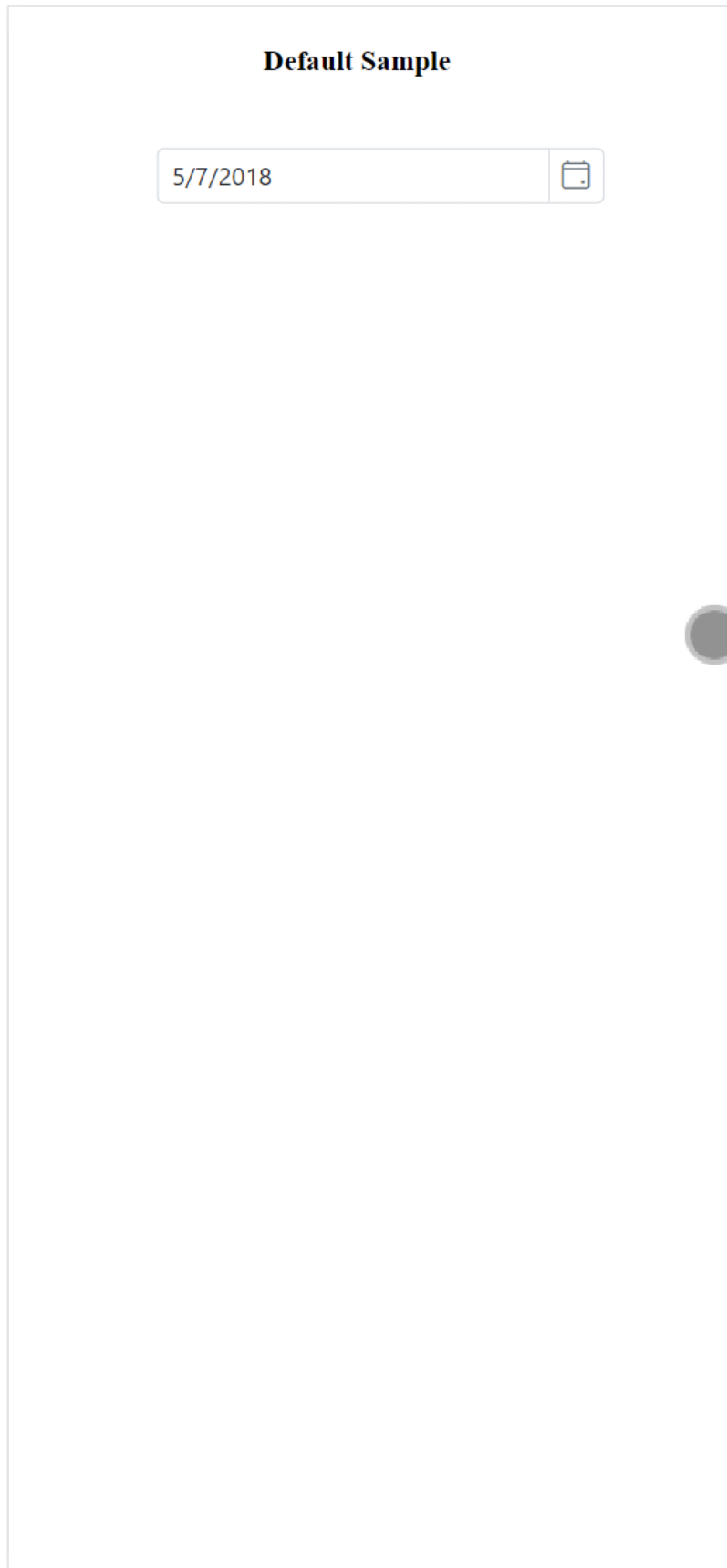
[Customizing Calendar's style and appearance](#)

Full screen mode support in mobiles and tablets

The DatePicker component's full-screen mode feature enables users to view the component popup element in full-screen mode on mobile devices with improved visibility and a better user experience. It is important to mention that this feature is exclusively available for mobile and tablet devices in both landscape and portrait orientations. To activate the full screen mode within the DatePicker component, simply set the [fullScreenMode](#) API value to `true`. This action will extend the calendar element to occupy the entire screen on mobile devices.

```
`javascript
import { Component } from '@angular/core';

@Component({
selector: 'app-root',
template: `<!-- To Render DatePicker -->
<ejs-datepicker [fullScreenMode]="true"></ejs-datepicker>`
})
export class AppComponent { }
`
```



How To

Json data binding in Angular Datepicker component

In most of the real cases, the model data will be available with JSON format only.

Here we have showcased DatePicker component by setting JSON string to value property.

In this JSON, we have used ISO formatted date string which is frequently used date format to get proper date and time value without any misreading.

Also our DatePicker component supports the ISO formatted date value, so parsed JSON value can be directly set to DatePicker model.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
export interface User {
    selectedDate: Date;
}
export interface JSONUser {
    selectedDate: string;
}
@Component({
    imports: [

        DatePickerModule //declaration of ej2-angular-calendars module into
NgModule
    ],
    standalone: true,
    selector: 'app-root',
    template: `
        <!-- To Render DatePicker -->
        <ejs-datepicker id="datepicker" width='245px'
        [(value)]='user.selectedDate' (change)='onChange($event)'></ejs-datepicker>
        <div class="valuestring">
            <b>User model</b>: <br/>{{user | json }}
            <br/><br/>
            <b>JSON Data</b>: <br/>{{ model_result }}
            <br/><br/>
        </div>`
    })
export class AppComponent {
    public user?: User|any;
    public JSONData: JSONUser = JSON.parse('{ "selectedDate": "2018-12-18T08:56:00+00:00"}');
    public model_result: string = JSON.stringify(this.JSONData);
    public ngOnInit() {
        this.user = this.JSONData;
    }
    onChange(args: any) {
        this.JSONData.selectedDate = args.value;
        this.model_result = JSON.stringify(this.JSONData);
    }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Two way binding in Angular DatePicker component

The following example demonstrates how to achieve **two-way binding** by binding the **value** to the first DatePicker component by using property binding and binding the model data using **ngModel** to the second DatePicker component. The **value** of the DatePicker will get change, when their is any change in the property value or model value.

The two-way binding can also be achieved only by using **property binding** or **model binding** in the DatePicker component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { CalendarModule } from '@syncfusion/ej2-angular-calendars'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    CalendarModule,
    DatePickerModule,
    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./style.css'],
  template: `
    <!-- two-way binding using the value binding and model binding in the
    DatePicker --->
    <ejs-datepicker id="fisrtdatepicker" #ejDatePicker [(value)]= 'value'
    width="230px"></ejs-datepicker>
    <ejs-datepicker id="seconddatepicker" #ejDatePickers
    [(ngModel)]= 'value' width="230px"></ejs-datepicker>
  `
})
export class AppComponent {
  value: Date;
  constructor() {
    this.value = new Date('1/1/2020');
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
```



```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Disable placeholder readonly in Angular Datepicker component

Property | Purpose

[enabled](#) | The component can be restricted on a page, by setting enabled value as false which will disable the component completely from all user interactions including in form post action.

[placeholder](#) | Using `placeholder` you can display a short hint in the input element.

[readonly](#) | set `readonly` to stop editing the value in the input, but value can be included when form submit.

The following example demonstrates how to achieve the above described properties in the DatePicker.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './template.html'
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Prevent the popup close in Angular Datepicker component

The following examples demonstrates how to prevent the popup from closing.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
import { PreventableEventArgs } from '@syncfusion/ej2-angular-calendars';
@Component({
  imports: [
    DatePickerModule
  ],
  standalone: true,
```

```

        selector: 'app-root',
        template: `<ejs-datepicker (close)='onClose($event)' placeholder='Enter
date'></ejs-datepicker>`
    })
    export class AppComponent {
        onClose(args: PreventableEventArgs | any): void {
            args.preventDefault();
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Open datepicker popup on input click in Angular Datepicker component

You can open the DatePicker popup on input focus by calling the `show` method in the input `focus` event.

The following example demonstrates how to open the DatePicker popup when the input is focused.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
import { CalendarComponent, FocusEventArgs } from '@syncfusion/ej2-angular-
calendars';
@Component({
    imports: [

        DatePickerModule
    ],
    standalone: true,
    selector: 'app-root',
    template: `<ejs-datepicker #default (focus)='onFocus($event)'
placeholder='Choose a date'></ejs-datepicker>`
})
export class AppComponent {
    @ViewChild('default')
    public datePickerObj?: CalendarComponent;
    onFocus(args: FocusEventArgs): void {
        (this.datePickerObj as any).show();
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Css customization in Angular Datepicker component

To customize DatePicker UI, you can make use of [cssClass](#) which will be added to DatePicker component as the root CSS class. With this CSS class, you can override existing styles of DatePicker.

Following is the list of classes that provides flexible way to customize the DatePicker component.

Class Name	Description
---	---
e-date-wrapper	Applied to DatePicker wrapper element.
e-datepicker	Applied to input element and DatePicker popup element.
e-datepicker .e-date-wrapper	Applied to input element only.
e-input-group-icon.e-date-icon	Applied to popup icon button.
e-float-text	Applied to floating label text element.
e-popup	Applied to popup element.
e-datepicker.e-popup	Applied to DatePicker popup element only.
e-header	Applied to Calendar header.
e-title	Applied to Calendar title.
e-icon-container	Applied to Calendar previous and next icon container.
e-prev	Applied to Calendar previous icon.
e-next	Applied to Calendar next icon.
e-weekend	Applied to Calendar weekend dates.
e-other-month	Applied to Calendar other month dates.
e-day	Applied to each day cell of the Calendar.
e-selected	Applied to Calendar selected dates.
e-disabled	Applied to Calendar disabled dates.
e-footer-container	Applied to the Calendar footer container.
e-today	Applied to the Calendar Today Cell.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./index.css'],
})
```

```

    template: `<ejs-datepicker [value]='dateValue' [cssClass]='cssClass'
placeholder='Enter date' (renderDayCell)='onRenderCell($event)'></ejs-
datepicker>`
  })
  export class AppComponent {
    onRenderCell($event: any) {
      throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Custom event emitter in Angular DatePicker component

The **two-way binding** in DatePicker can also be achieved using the custom event binding and property binding in the controls present in two different components. To create custom event, we need to create an instance of **event emitter**.

In the following example, **property binding** is used to share the data from the parent component to the child component using **@input** directive and **custom event binding** is used to share the data from the child component to the parent component by using **@output** directive.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
import { DatePickerComponent } from '@syncfusion/ej2-angular-calendars';
@Component({
  imports: [ DatePickerModule ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div class="parentelement">
      <div class="datevalue">
        <ejs-datepicker id="datepicker" #date (change)="deposit()"
placeholder="Parent component" floatLabelType="Always" [value]="value"
width="200px"></ejs-datepicker>
      </div>
    </div>
    <child [xvalue]="value" (valueChange)="valuecheck($event)"> </child>
  `,
})
export class ParentComponent {
  @ViewChild('date')
  public Date?: DatePickerComponent;
  value: Date;
  constructor() {
```

```

        this.value = new Date("2/1/2020");
    }
    deposit() {
        this.value = this.Date?.value as Date;
    }
    valuecheck(args: any) {
        this.value = args;
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Custom validation using form validator in Angular DatePicker component

The client side validation takes place in the browser to avoid the waiting time to receive the response from sever. It validates the form elements to provide the better feedback messages to correct the every fields before the form submission.

To achieve the client side validation in a DatePicker component by using **FormValidator** function. It provides an option to customize the feedback error messages to the corresponding fields to take action to resolve the issue.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild, OnInit } from '@angular/core';
import { DatePickerComponent } from '@syncfusion/ej2-angular-calendars';
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
@Component({
  imports: [

    DatePickerModule,
    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<form id="form-element" class="form-vertical">
    <ejs-datepicker #ejDate id='datepicker' placeholder='Enter date'
width="275px"(blur)="onFocusOut()" (change)= "onChange($event)"></ejs-
datepicker>
    </form>`
})
export class AppComponent implements OnInit {
  @ViewChild('formElement') element: any;
  @ViewChild('ejDate') ejDate?: DatePickerComponent;
  public formObject?: FormValidator;
  ngOnInit() {
    // custom validator function.

```

```

    let customFn: (args: {
      [key: string]: string
    }) => boolean = (args: {
      [key: string]: string
    }) => {
      return (((this.ejDate as DatePickerComponent)
        ).value).getFullYear() > 1990 && (((this.ejDate as DatePickerComponent)
        ).value).getFullYear() < 2020);
    };
    let options: FormValidatorModel = {
      rules: {
        'datepicker': {
          required: [true, "Value is required"]
        }
      },
      customPlacement: (inputElement: HTMLElement, errorElement:
        HTMLElement) => {
        inputElement.parentElement?.parentElement?.appendChild(errorElement);
      }
    };
    this.formObject = new FormValidator('#form-element', options);
    this.formObject.addRules('datepicker', {
      range: [customFn, "Please select a date between years from 1990
to 2020"]
    });
    this.formObject = new FormValidator('#form-element', options);
  }
  // Form validation takes place when focus() event of DatePicker is
  // triggered.
  public onFocusOut(): void {
    this.formObject?.validate("datepicker");
  }
  // Custom validation takes place when value is changed.
  public onChange(args: any) {
    if (this.ejDate?.value != null)
      this.formObject?.validate("datepicker");
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Reactive form in Angular DatePicker component

DatePicker is a form component and validation is playing vital role in forms to get the valid data.

Here to showcase the DatePicker with form validations we have used the reactive form.

- The reactive forms uses the reactive model-driven technique to handle form data between component and view, due to that we also call it as the model-driven forms.

It's listen the form data changes between App component and view, also it returns the valid states and values of form elements.

For more details about Reactive Forms refer: <https://angular.io/guide/reactive-forms>.

- For the reactive forms, import a **ReactiveFormsModule** into app module as well as the **FormGroup**,

FormControl should be imported to app component.

The **FormGroup** is used to declare **formGroupName** for the form and the **FormControl** is used to declare **formControlName** for form controls. Declare the **formControlName** to DatePicker component as usual.

Then, create a value object to the **FormGroup** and each value will be the default value of the form control.

The following example demonstrates how to use the reactive forms.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewChild, OnInit, ElementRef, Inject } from
 '@angular/core';
import { DatePickerComponent } from '@syncfusion/ej2-angular-calendars';
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { FormsModule } from '@angular/forms';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, DatePickerModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './template.html'
})
export class AppComponent implements OnInit {
  @ViewChild('ejDatePicker') ejDatePicker?: DatePickerComponent;
  public targetElement?: HTMLElement;
  public placeholder: String = 'Date of Birth';
  skillForm?: FormGroup | any;
  build: FormBuilder;
  constructor(@Inject(FormBuilder) private builder: FormBuilder) {
    this.build = builder;
    this.createForm();
  }
  ngOnInit(): void {
    throw new Error('Method not implemented.');
```

```

        username: ['', Validators.required],
        usermail: ['', Validators.email],
    });
}
get username() {
    return (this.skillForm as FormGroup<any> | any).get('username');
}
get datepicker() {
    return (this.skillForm as FormGroup<any> | any).get('datepicker');
}
get usermail() {
    return (this.skillForm as FormGroup<any> | any).get('usermail');
}
onSubmit() {
    alert("Form Submitted!");
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Template driven forms in Angular DatePicker component

The form can be build with Angular template syntax easily along with form specific directives.

This template-drive forms uses the `ng` directives in view to handle the forms controls.

- To enable the template-driven, import the FormsModule into corresponding app component.

For more details about template-driven Forms refer to: <https://angular.io/guide/forms#template-driven-forms>.

- In angular forms mentioning the name is must to process as form elements.
- Mention the `name` attribute to DatePicker element which will be used to identify the form element. To register an DatePicker element to ngForm, give the ngModel to it so the FormsModule will automatically detect the DatePicker as a form element.

After that, the DatePicker value will be selected based on the ngModel value.

APP.COMPONENT.TS

```

import { BrowserModule } from '@angular/platform-browser'
import { NgModule } from '@angular/core'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild, ViewEncapsulation, Inject } from '@angular/core';
import { FormGroup, FormBuilder, Validators } from '@angular/forms';
import { CalendarComponent } from '@syncfusion/ej2-angular-calendars';
class User {

```



```

    public date?: Date
    constructor() {

    }
}
@Component({
  imports: [ FormsModule, ReactiveFormsModule, DatePickerModule],
  standalone: true,
  selector: 'app-root',
  templateUrl: './template.html'
})
export class DefaultDatePickerComponent {
  user?: User | any;
  ngOnInit() {
    this.user = new User();
  }
  onSubmit(userForm: any) {
    (userForm.valid) ? alert("submitted") : alert("form is invalid");
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

TEMPLATE.HTML

```

<div>
  <form #userForm="ngForm" (ngSubmit)="onSubmit(userForm)">
    <div class="form-group">
      <ejs-datepicker id="datepicker" name="date" [(ngModel)]="user.date"
      #date="ngModel" width="300px" required></ejs-datepicker>
      <div *ngIf="userForm.invalid && (userForm.dirty ||
      userForm.touched)" class="alert alert-danger formerror">
        Valid date is required
      </div>
      <div *ngIf="userForm.valid && (userForm.dirty || userForm.touched)"
      class="alert alert-success formerror">
        <table>
          <tr>
            <td style="width:50%">Selected value: </td>
            <td class="formtext ">{{ user.date | date:"dd/MM/yyyy"}}</td>
          </tr>
        </table>
      </div>
    </div>
    <div class="buttons">
      <button type="submit" class="e-btn e-success">Submit</button>
      <button type="reset" class="e-btn e-warning">Reset</button>
    </div>
  </form>
  <div class="dataclass">userForm.value: {{userForm.value | json}}</div>
  <div class="dataclass">userForm.valid: {{userForm.valid}}</div>

```

```

</div>
<style>
  form {
    margin-top: 50px;
  }
</style>

```

Customize the datepicker day header in Angular Datepicker component

You can change the format of the day that to be displayed in header using [dayHeaderFormat](#) property. By default, the format is **Short**.

You can find the possible formats on below.

Name	Description

Short	Sets the short format of day name (like Su) in day header.
Narrow	Sets the single character of day name (like S) in day header.
Abbreviated	Sets the min format of day name (like Sun) in day header.
Wide	Sets the long format of day name (like Sunday) in day header.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, ViewChild } from '@angular/core';
import { DatePickerComponent } from '@syncfusion/ej2-angular-calendars';
import { DropDownListComponent, ChangeEventArgs } from '@syncfusion/ej2-
angular-dropdowns';
@Component({
  imports: [

    DropDownListModule,
    DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./style.css'],
  template: `
    <div id="container">
      <div id="datepicker">
        <ejs-datepicker #default dayHeaderFormat='Short'></ejs-
datepicker>
      </div>
      <div id="format">
        <label class="custom-input-label">Header Format Types</label>
        <div id="wrap">
          <ejs-dropdownlist id="dayformat" #select
[dataSource]='formatData' [(value)]='value' [fields]='fields'
[placeholder]='waterMark' (change)='formatHandler($event)'></ejs-
dropdownlist>

```

```

        </div>
    </div>
</div>
}))
export class AppComponent {
    @ViewChild('default')
    public datePickerObj?: DatePickerComponent;
    @ViewChild('select')
    public dayHeaderFormat?: DropDownListComponent;
    // define the JSON of data
    public formatData: Object[] = [
        { Id: 'Short', Label: 'Short' },
        { Id: 'Narrow', Label: 'Narrow' },
        { Id: 'Abbreviated', Label: 'Abbreviated' },
        { Id: 'Wide', Label: 'Wide' },
    ];
    public fields: Object = { text: 'Label', value: 'Id' };
    public waterMark: string = 'Select format type';
    public value: string = 'Short';
    public formatHandler(args: ChangeEventArgs): void {
        (this.datePickerObj as DatePickerComponent).dayHeaderFormat =
args.value as any;
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Ej1 api migration in Angular Datepicker component

This article describes the API migration process of DatePicker component from Essential JS 1 to Essential JS 2.

Date Selection

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Setting the value	Property: value	Property: value

Date Format

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Display date format	Property: dateFormat	Property: format
Day header format	Property: dayHeaderFormat	Not Applicable

Calendar Views

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Start view	Property: startLevel	Property: start
Depth view	Property: depthLevel	Property: depth

Date Range

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Minimum date	Property: minDate	Property: min
Maximum date	Property: maxDate	Property: max

Disabled Dates

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Block-out dates	Property: blackoutDates blackoutDates : Object = [new Date(2016, 4, 10), new Date(2016, 4, 15), new Date(2016, 4, 20), new Date(2016, 4, 22), new Date(2016, 5, 12), new Date(2016, 5, 24)]	Can be achieved by public disableDate(args) { if (args.date.getDay() === 0 args.date.getDay() === 6) { args.isDisabled = true; } }

Customization

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
CSS Class	Property: cssClass	Property: cssClass
Event callback for each cell creation	Not Applicable	Event: renderDayCellpublic onRenderCell(args:any) {/ code block */}
Show/Hide the today button	Property: showFooter	Property: showTodayButton
Show/Hide the other month dates	Property: showOtherMonths	Can be achieved bycss.e-datepicker .e-calendar .e-content tr.e-month-hide, .e-datepicker .e-calendar .e-content td.e-other-month > .e-day { visibility: none; }.e-datepicker .e-calendar .e-content td.e-month-hide, .e-datepicker .e-calendar .e-content td.e-other-month { pointer-events: none; touch-action: none;}
Show/Hide the disabled date	Property: showDisabledRangeblackoutDates : Object = [new Date(2016, 4, 10), new Date(2016, 4, 15), new Date(2016, 4, 20), new Date(2016, 4, 22), new Date(2016, 5, 12), new Date(2016, 5, 24)]	Not Applicable
Show/Hide the popup button	Property: showPopupButton	Can be achieved by@ViewChild("dateObj") dateObj: DatePickerComponent;public onFocus(args:any) { this.dateObj.show();}css.e-control-wrapper .e-input-group-icon.e-date-icon { display: none;
Enable/Disable the rounded corner	Property: showRoundedCorner	Can be achieved bycss.e-control-wrapper.e-customStyle.e-date-wrapper.e-input-group { border-radius: 4px;}
Skip a month	Property: stepMonths	Can be achieved by@ViewChild("dateObj") dateObj: DatePickerComponent;public onOpen(args:any){ this.dateObj.navigateTo('Year', new Date("03/18/2028"));
Show/Hide the tooltip	Property: showTooltip	Not Applicable

Today button text	Property: buttonText	Can be achieved by <code>L10n.load({ 'de': { 'datepicker': { placeholder: 'Wählen Sie ein Datum aus', today: 'heute' } } });</code>
Display inline	Property: displayInline	Not Applicable
Enable/Disable the animation	Property: enableAnimation	Not Applicable
Highlight dates	Property: highlightSection	Can be achieved by <code>public highlightDate(args:any) { if (args.date.getDate() === 10) { args.element.classList.add('e-highlightweekend'); } } css.e-highlightweekend { background-color: #cfe9f3; }</code>
Highlight weekend	Property: highlightWeekend	Can be achieved by <code>public highlightDate(args:any) { if (args.date.getDay() === 0 args.date.getDay() === 6) { args.element.classList.add('e-highlightweekend'); } } css.e-highlightweekend { background-color: #cfe9f3; }</code>
Tooltip format	Property: tooltipFormat	Not Applicable
Special dates	Property: specialDates specialDate: Object = [{ date: new Date(), tooltip: "In Australia" }]	Can be achieved by <code>public customDates(args:any) { let span: HTMLElement; if (args.date.getDate() === 10) { span = document.createElement('span'); span.setAttribute('class', 'e-icons highlight'); args.element.firstChild.setAttribute('title', 'Birthday !'); args.element.setAttribute('title', 'Birthday !'); args.element.setAttribute('data-val', 'Birthday !'); args.element.appendChild(span); } }</code>
FocusIn event	Event: focusIn <code>public onFocus(e:any) { / code block */ }</code>	Event: focus <code>public onFocus(args:any) { / code block */ }</code>
FocusOut event	Event: focusOut <code>public onFocus(e:any) { / code block */ }</code>	Event: blur <code>public onBlur(args:any) { / code block */ }</code>
FocusIn method	Not Applicable	Method: focusIn() <code>@ViewChild("dateObj") dateObj: DatePickerComponent; public create(args:any) { this.dateObj.focusIn(); }</code>

FocusOut method	Not Applicable	Method: focusOut() @ViewChild("dateObj") dateObj : DatePickerComponent; public create(args:any){ this.dateObj.focusIn(); this.dateObj.focusOut();}
Prevent popup close	Not Applicable	Event: Close public onClose(args) { /Triggers when the popup gets close/ args.cancel = true;}
Prevent popup open	Not Applicable	Event: Open public onOpen(args:any) { /Triggers when the popup gets close/ args.cancel = true;}

Accessibility

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Enable/Disable the RTL	Property: enableRTL	Property: enableRtl

Persistence

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Enable/Disable the persistence	Property: enablePersistence	Property: enablePersistence

Validation

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Validation rules	Property: validationRules:validationRules:Object;constructor(){ this.validationRules = {required:true};}	Can be achieved bylet options: FormValidatorModel = { rules: { 'datepicker': { required: [true, "Value is required"] } } }; this.formObject = new FormValidator('#form-element', options);
Validation message	Property: validationMessage:validationRules:Object;validationMessage:Object;constructor(){ this.validationMessage = {required: "Required Date value"}; this.validationRules = {required:true};}	Can be achieved bylet options: FormValidatorModel = { rules: { 'datepicker': { required: [true, "Value is required"] } }, customPlacement: (inputElement: HTMLElement, errorElement: HTMLElement) => { inputElement.parentElement.parentElement.appendChild(errorElement); } };this.formObject = new FormValidator('#form-element', options);

Common

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Width	Property: width	Property: width
Readonly	Property: readonly	Property: readonly
Show/Hide the clear button	Not Applicable	Property: showClearButton
Height	Property: height	Can be achieved by <code>css.e-control-wrapper.e-custom-style.e-date-wrapper.e-input-group { height: 35px;}</code>
Html attributes	Property: <code>HtmlAttributes</code> <code>htmlAttributes : Object = {required:"required"}</code>	Not Applicable
Enable/Disable the week number	Property: weekNumber	Property: weekNumber
Watermark text	Property: watermarkText	Property: placeholder
Disable/Enable	Property: enabled	Property: enabled
Disable the DatePicker	Method: <code>disable()</code> <code>public onCreate(e:any) { var dateObject = \$("#datepicker").data("ejDatePicker"); dateObject.disable();}</code>	Not Applicable
Enable the DatePicker	Method: <code>enable()</code> <code>public onCreate(e:any) { var dateObject = \$("#datepicker").data("ejDatePicker"); dateObject.enable();}</code>	Not Applicable
Enable/Disable the textBox editing	Property: AllowEdit	Property: AllowEdit
zIndex	Not Applicable	Property: zIndex
Specify the placeholder text behavior	Not Applicable	Property: floatLabelType

Globalization

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Locale	Property: Locale	Property: locale
First day of week	Property: startDay	Property: firstDayOfWeek

Strict Mode

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Strict mode	Property: enableStrictMode	Property: strictMode

Open and Close

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Close	Event: Closepublic onClose(e:any) { / code block */ }	Event: Closepublic onClose(args:any) { / code block */ }
Hide	Method: hide()public onCreate(e:any) { var dateObject = \$("#datepicker").data("ejDatePicker"); dateObject.show(); dateObject.hide();}	Method: hide()@ViewChild("dateObj") dateObj: DatePickerComponent;public onCreate(args:any){ this.dateObj.show(); this.dateObj.hide();}
Open	Event: Openpublic onOpen(e:any) { / code block */ }	Event: Openpublic onOpen(args:any) { / code block */ }
Show	Method: show()public onCreate(e:any) { var dateObject = \$("#datepicker").data("ejDatePicker"); dateObject.show();}	Method: show()@ViewChild("dateObj") dateObj: DatePickerComponent;public onCreate(args:any){ this.dateObj.show();}

View Navigation

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Navigate to specific month	Not Applicable	Method: navigateTo()@ViewChild("dateObj") dateObj: DatePickerComponent;public onOpen(args:any){ this.dateObj.navigateTo('Year', new Date("03/18/2028"));
Navigation callback	Event: Navigatepublic onNavigate(e:any) {/ code block */ }	Event: Navigatedpublic onNavigated(args:any) {/ code block */ }
Enable/Disable the drill down	Property: allowDirllDown	Not Applicable

DateRangePicker

Getting started with Angular Daterangepicker component

The following section explains the steps required to create a simple DateRangePicker component and also it demonstrates the basic usage of the DateRangePicker.

Dependencies

Install the below required dependency packages in order to use the `DateRangePicker` component in your application.

```
`javascript
|-- @syncfusion/ej2-angular-calendars
|-- @syncfusion/ej2-angular-base
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-calendars
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-splitbuttons
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-buttons
`,`
```

Setup Angular environment

Angular provides the easiest way to set angular CLI projects using [Angular CLI](#) tool.

Install the CLI application globally to your machine.

```
`bash
npm install -g @angular/cli
`,`
```

Create a new application

```
`bash
ng new syncfusion-angular-daterangepicker
`,`
```

By default, it install the CSS style base application. To setup with SCSS, pass `--style=scss` argument on create project.

Example code snippet.

```
`bash
ng new syncfusion-angular-daterangepicker --style=scss
`,`
```

Navigate to the created project folder.

```
`bash
cd syncfusion-angular-daterangepicker
`
```

Installing Syncfusion DateRangePicker package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages($\geq 20.2.36$) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-calendars](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-calendars --save
`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-calendars@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-calendars@ngcc --save
`
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-calendars:"20.2.38-ngcc"
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Registering DateRangePicker module

Import DateRangePicker module into Angular application(`src/app/app.module.ts`) from the package `@syncfusion/ej2-angular-calendars`.

```
`javascript
import { NgModule }    from '@angular/core';
```

```
import { BrowserModule } from '@angular/platform-browser';
// import the DateRangePickerModule for the DateRangPicker component
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars';
import { AppComponent } from './app.component';
@NgModule({
  //declaration of DateRangePickerModule into NgModule
  imports: [ BrowserModule, DateRangePickerModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Adding CSS reference

The following CSS files are available in `../node_modules/@syncfusion` package folder..

This can be referenced in [src/styles.css] using following code.

```
`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';
@import '../node_modules/@syncfusion/ej2-angular-calendars/styles/material.css';
```

If you want to refer the combined component styles, please make use of our [CRG](#) (Custom Resource Generator) in your application.

Adding DateRangePicker component

Modify the template in [src/app/app.component.ts] file to render the DateRangePicker component. by using `<ejs-daterangepicker>` selector.

```
`javascript
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
```

```

template: `<!-- To Render DateRangePicker -->
<ejs-daterangepicker></ejs-daterangepicker>`
})
export class AppComponent { }
`

```

Running the application

After completing the configuration required to render a basic DateRangePicker, run the following command to display the output in your default browser.

```
ng serve
```

The following example illustrates the output in your browser

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule,
    DateRangePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-daterangepicker></ejs-daterangepicker>`
})
export class AppComponent {
  constructor() {
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Setting the start and end date

The start and end date in a range can be set by using `startDate` and `endDate` properties. To know more about range restriction in DateRangePicker, please refer this [page](#).

The below example demonstrates the DateRangePicker with startDate and endDate.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [

    FormsModule,
    DateRangePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-daterangepicker placeholder='Select a range'
[startDate]='start' [endDate]='end'></ejs-daterangepicker>`
})
export class AppComponent {
  public month: number = new Date().getMonth();
  public fullYear: number = new Date().getFullYear();
  public start: Date = new Date(this.fullYear, this.month - 1, 7);
  public end: Date = new Date(this.fullYear, this.month, 25);
  constructor() {
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

You can refer to our [Angular Date Range Picker](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Date Range Picker example](#) that shows how to render the Date Range Picker in Angular.

See Also

- [Render DateRangePicker with pre-defined ranges](#)
- [Render DateRangePicker with specific culture](#)
- [How to achieve validation with DateRangePicker](#)
- [How to achieve two-way binding with DateRangePicker](#)
- [Reactive forms with DateRangePicker](#)

Range selection in Angular Daterangepicker component

Range selection in a DateRangePicker can be made-to-order with desire restrictions based on application needs.

Restrict the range within a range

You can restrict the minimum and maximum date that can be allowed as start date, end date in a range selection with help of [min](#), [max](#) properties.

- **min** – sets the minimum date that can be selected as startDate.
- **max** – sets the maximum date that can be selected as endDate

In below sample, you can select a range from 15th day of this month to 15th day of next month.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [

    FormsModule,
    DateRangePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./index.css'],
  template: `<ejs-daterangepicker [min]='minDate' [max]='maxDate'
placeholder='Select a range'></ejs-daterangepicker>`
})
export class AppComponent {
  public today: Date = new Date();
  public currentYear: number = this.today.getFullYear();
  public currentMonth: number = this.today.getMonth();
  public currentDay: number = this.today.getDate();
  public minDate: Object = new Date(this.currentYear, this.currentMonth,
15);
  public maxDate: Object = new Date(this.currentYear, this.currentMonth+1,
15);
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

If the value of **min** or **max** properties changed through code behind, then you have to update the **start date**, **end date** property to set within the range. Or else , if the **start** and **end date** is out of specified date range, a validation error class will be appended to the input element. If **strictMode** is enabled, and both the start, end date is lesser than the min date then start and end date will be updated with **min** date. If both the start and end date is higher than the max date then start and end date will be updated with **max** date. Or else, if startDate is less than **min** date, startDate will be updated with **min** date or if endDate is greater than **max** date, endDate will be updated with the **max** date.

Range span

Span between ranges can be limited in order to avoid excess or less days selection towards the required days in a range.

In this, minimum and maximum span that can be allowed within the date range can be customized by [minDays](#), [maxDays](#) properties.

- **minDays**- Sets the minimum number of days between start date and end date.
- **maxDays**- Sets the maximum number of days between start date and end date.

In below sample, the range selection should be greater than 3 days and less than 8 days else it won't set.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule,
    DateRangePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./index.css'],
  template: `<ejs-daterangepicker [minDays]='minDays' [maxDays]='maxDays'
placeholder='Select a range'></ejs-daterangepicker>`
})
export class AppComponent {
  public minDays: Number = 4;
  public maxDays: Number = 7;
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Strict mode

DateRangePicker provides the option to limit the user towards entering the valid date only.

With strict mode, you can set only valid selection. Also, If any invalid range is specified then the date range value will reset to previous value.

This restriction can be availed by enabling the [strictMode](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule,
    DateRangePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./index.css'],
  template: `<ejs-daterangepicker strictMode='true' placeholder='Select a range'></ejs-daterangepicker>`
})
export class AppComponent {
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Globalization in Angular Daterangepicker component

Globalization is the combination of internalization and localization. You can adapt the component to various languages by parsing and formatting the date or number [Internationalization](#) and also add culture specific customization and translation to the text [localization](#).

By default, DateRangePicker date format, week, and month names are specific to the English culture. It utilizes the [Essential JavaScript 2 Internationalization](#) package to parse and format the date object based on the culture by using the official [UNICODE CLDR](#) JSON data. It provides the `loadCldr` method to load the culture specific CLDR JSON data. To go with the different culture other than English, follow the below steps.

- Install the `CLDR-Data` package by using the below command (it installs all the CLDR JSON data).
To

know about CLDR-Data refer the [CLDR-Data](#) link.

,

```
npm install cldr-data --save
```

,

Once the package installed, you can find the culture specific JSON data under the location `/node_modules/cldr-data`.

- Now import the installed CLDR JSON data into the `app.component.ts` file.
- Now use the [loadCldr](#) method to load the culture specific CLDR JSON data

from the installed location to `app.component.ts` file.

- DateRangePicker displayed **Sunday** as the first day of week based on default culture ("en-US"). If you want to display the DateRangePicker with loaded culture's first day of week, you need to import `weekdata.json` file from the `cldr-data/supplemental` as given in the code example.

```
`typescript
```

```
import { loadCldr } from '@syncfusion/ej2-base';
declare var require: any;
loadCldr(
  require('cldr-data/supplemental/numberingSystems.json'),
  require('cldr-data/main/de/ca-gregorian.json'),
  require('cldr-data/main/de/numbers.json'),
  require('cldr-data/main/de/timeZoneNames.json'),
  require('cldr-data/supplemental/weekdata.json') // To load the culture based first day of week
);
`
```

The **Localization** library allows you to localize default text content of the DateRangePicker. The DateRangePicker component has static text for **today** feature that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

Locale keywords | Text

placeholder | Hint to describe expected value in input element.

startLabel | Label to represent the start date.

endLabel | Label to represent the end date.

applyText | Text present in the apply button.

cancelText | Text present in the cancel button.

selectedDays | Text to represent selected days.

days | Text represents days.

customRange | Text present in the custom range button in presets container.

- Before changing a culture other than **English**, ensure that locale text for the concern culture is loaded through `L10n.load` method.

```
`typescript
```

```
//Load the L10n from ej2-base
import { L10n } from '@syncfusion/ej2-base';
//load the locale object to set the localized placeholder value
L10n.load({
  'de': {
    'daterangepicker': { placeholder: 'Wählen Sie einen Bereich aus',
      startLabel: 'Wählen Sie Startdatum',
      endLabel: 'Wählen Sie Enddatum',
      applyText: 'Sich bewerben',
      cancelText: 'Stornieren',
      selectedDays: 'Ausgewählte Tage',
      days: 'Tage',
      customRange: 'benutzerdefinierten Bereich'
    }
  }
});
`
```

- Set the culture by using the [locale](#) property. In this below code example, initialize the DateRangePicker component in **German** culture with corresponding localized text.

```
`typescript
import { Component } from '@angular/core';
import { L10n, loadCldr } from '@syncfusion/ej2-base';
declare var require: any;
loadCldr(
  require('cldr-data/supplemental/numberingSystems.json'),
  require('cldr-data/main/de/ca-gregorian.json'),
  require('cldr-data/main/de/numbers.json'),
  require('cldr-data/main/de/timeZoneNames.json')
);
@Component({
  selector: 'app-root',
  template: `
    <ejs-daterangepicker locale='de'></ejs-daterangepicker>
  `
})
```

```

,
})
export class AppComponent {
  constructor() {
  }
  ngOnInit(): void {
    L10n.load({
      'de': {
        'daterangepicker': {
          placeholder: 'Wählen Sie einen Bereich aus',
          today: "heute"
          startLabel: 'Wählen Sie Startdatum',
          endLabel: 'Wählen Sie Enddatum',
          applyText: 'Sich bewerben',
          cancelText: 'Stornieren',
          selectedDays: 'Ausgewählte Tage',
          days: 'Tage',
          customRange: 'benutzerdefinierten Bereich'
        }
      }
    });
  }
},

```

The following sample demonstrate the DateRangePicker component in **German** culture.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
import { loadCldr, L10n } from '@syncfusion/ej2-base';
// Here we have referred local json files for preview purpose
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as detimeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, detimeZoneNames);
@Component({

```

```

imports: [
    DateRangePickerModule
],
standalone: true,
selector: 'app-root',
template: `<ejs-daterangepicker locale='de'></ejs-daterangepicker>`
})
export class AppComponent {
    constructor() {
    }
    ngOnInit(): void {
        L10n.load({
            'de': {
                'daterangepicker': { placeholder: 'Wählen Sie einen Bereich
aus',
                    today:"heute",
                    startLabel: 'Wählen Sie Startdatum',
                    endLabel: 'Wählen Sie Enddatum',
                    applyText: 'Sich bewerben',
                    cancelText: 'Stornieren',
                    selectedDays: 'Ausgewählte Tage',
                    days: 'Tage',
                    customRange: 'benutzerdefinierten Bereich'
                }
            }
        });
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Right-To-Left

The DateRangePicker supports RTL (right-to-left) functionality for languages like Arabic, Hebrew to display the text in the right-to-left direction. Use [enableRtl](#) property to set the RTL direction.

The below code example demonstrates the DateRangePicker component in Hebrew culture, also explains how to set the localized text to the placeholder using `L10n.load` method.

`typescript

```

import { Component } from '@angular/core';
import { L10n, loadCldr } from '@syncfusion/ej2-base';
declare var require: any;
loadCldr(
    require('cldr-data/supplemental/numberingSystems.json'),
    require('cldr-data/main/he/ca-gregorian.json'),

```

```

require('cldr-data/main/he/numbers.json'),
require('cldr-data/main/he/timeZoneNames.json')
);
@Component({
  selector: 'app-root',
  template: <ejs-daterangepicker locale='he' [enableRtl]='isRTL'></ejs-daterangepicker>
})
export class AppComponent {
  public isRTL: boolean = true;
  constructor() {
  }
  ngOnInit(): void {
    L10n.load({
      'he': {
        'daterangepicker': { placeholder: 'בחר טווח',
          today: 'היום',
          startLabel: 'תוויית התחלה',
          endLabel: 'ח',
          applyText: 'להחיל טקסט',
          cancelText: 'בטל טקסט',
          selectedDays: 'ימים נבחרים',
          days: 'אִימים',
          customRange: 'טווח מותאם אישית'
        }
      }
    });
  }
}

```

The following example demonstrates the DateRangePicker in **Hebrew** culture with right-to-left direction.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'

```

```

import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
import { loadCldr, L10n } from '@syncfusion/ej2-base';
// Here we have referred local json files for preview purpose
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as hetimeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, hetimeZoneNames);
@Component({
  imports: [

    DateRangePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-daterangepicker locale='he' [enableRtl]='isRTL'></ejs-daterangepicker>`
})
export class AppComponent {
  public isRTL: boolean = true;
  constructor() {
  }
  ngOnInit(): void {
    L10n.load({
      'he': {
        'daterangepicker': { placeholder: 'בחר טווח',
          today: 'היום',
          startLabel: 'תווית התחלה',
          endLabel: 'סוף',
          applyText: 'להחיל טקסט',
          cancelText: 'בטל טקסט',
          selectedDays: 'ימים נבחרים',
          days: 'ימים',
          customRange: 'טווח מותאם אישית'
        }
      }
    });
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Date format customization

Representation of start and end date strings can be customized to required format by using format property.

By default, it will set based on the culture.

To know more about the date format standards, refer to the Internationalization Date Format section.

In below sample, the date strings are formatted to `yyyy-MM-dd` and separator between the date ranges is set to "to".

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule,
    DateRangePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-daterangepicker [format]='dateFormat' separator='to'
placeholder='Select a range'></ejs-daterangepicker>`
})
export class AppComponent {
  public dateFormat: String = "yyyy-MM-dd";
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customization in Angular Daterangepicker component

DateRangePicker makes available for the UI customization which can be achieved with properties, events that are available with this component.

Day cell format

[renderDayCell](#) is a event which provides the option to customize each day cell while rendering itself.

The following example disables the weekends of every month using `renderDayCell` event.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule,
    DateRangePickerModule
  ],
```

```

standalone: true,
  selector: 'app-root',
  template: `<ejs-daterangepicker (renderDayCell)= 'disableDate($event)'
placeholder='Select a range'></ejs-daterangepicker>`
})
export class AppComponent {
  constructor() {
  }
  disableDate(args: any): void {
    if (args.date.getDay() == 0 || args.date.getDay() == 6) {
      //sets isDisabled to true to disable the date.
      args.isDisabled = true;
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

First day of week

Start day in a week will differ based on culture and it can be customized based on application needs.

For this customization, you can make use of `firstDayOfWeek` property.

By default, first day of week in en-US is Sunday.

In below example, first day of the week in the pop-up calendar is customized to Monday with help of the `firstDayOfWeek` property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule,
    DateRangePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./index.css'],
  template: `<ejs-daterangepicker [firstDayOfWeek]='startDay'
placeholder='Select a range'></ejs-daterangepicker>`
})
export class AppComponent {
  public startDay: Number = 1;
  constructor() {
  }
}

```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Preset Ranges

DateRangePicker has an option to set the pre-defined ranges with label using **presets** property.

With help of this, we can set the frequently used ranges as preset ranges for quick selection in a DateRangePicker popup.

Here in following sample, you can choose the mostly using options from pre-defined ranges easily.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule,
    DateRangePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./index.css'],
  template: `<ejs-daterangepicker placeholder='Select a range'>
    <e-presets>
      <e-preset label="This Week" [start]='weekStart'
[end]='weekEnd'></e-preset>
      <e-preset label="This Month" [start]='monthStart'
[end]='monthEnd'></e-preset>
      <e-preset label="Last Month" [start]='lastStart'
[end]='lastEnd'></e-preset>
      <e-preset label="Last Year" [start]='yearStart'
[end]='yearEnd'></e-preset>
    </e-presets>
  </ejs-daterangepicker>`
})
export class AppComponent {
  public today: Date = new Date(new Date().toDateString());
  public weekStart: Date = new Date(new Date(new Date().setDate(new
Date().getDate() - (new Date().getDay() + 7) % 7)).toDateString());
  public weekEnd: Date = new Date(new Date(new Date().setDate(new Date(new
Date().setDate((new Date().getDate()
- (new Date().getDay() + 7) % 7)).getDate() + 6)).toDateString())
  ;
  public monthStart: Date = new Date(new Date(new
Date().setDate(1)).toDateString());
```

```

    public monthEnd: Date = this.today;
    public lastStart: Date = new Date(new Date(new Date(new
Date().setMonth(new Date().getMonth() - 1)).setDate(1)).toDateString());
    public lastEnd: Date = this.today;
    public yearStart: Date = new Date(new Date(new Date().setDate(new
Date().getDate() - 365)).toDateString());
    public yearEnd: Date = this.today;
    constructor() {
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [How to customize DateRangePicker using cssClass](#)
- [How to disable DateRangePicker component](#)
- [How to customize the DateRangePicker day header](#)

Accessibility in Angular Daterangepicker component

The DateRangePicker component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the DateRangePicker component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

```
| Accessibility Checker Validation |  |
| Axe-core Accessibility Validation |  |

<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The web accessibility makes web content and web applications more accessible for disabled people.

It especially helps in dynamic content change and development of advanced user interface controls with AJAX, HTML, JavaScript, and related technologies.

DateRangePicker provides built-in compliance with [WAI-ARIA](#) specifications. WAI-ARIA supports is achieved through the attributes like `aria-expanded`, `aria-disabled`, `aria-activedescendant` applied to the input element.

To know about the accessibility of Calendar refer to the Calendar's [Accessibility](#) section.

It helps disabled persons by providing information about the widget for assistive technology in the screen readers.

DateRangePicker component contains grid role and grid cell for each day cell.

- **Aria-expanded:** attributes indicates the state of a collapsible element.
- **Aria-disabled:** Indicates the disabled state of the DateRangePicker component.

Keyboard Interaction

You can use the following keys to interact with the DateRangePicker.

This component implements the keyboard navigation support by following the [WAI-ARIA practices](#).

It supports the following list of shortcut keys:

Input Navigation

Before opening the popup, use the below list of keys to control the popup element.

| **Press** | **To do this** |

| --- | --- |

| Alt + Down Arrow | Opens the popup. |

| Alt + Up Arrow | Closes the popup (if component's input element is in focused state). |

| Esc | closes the popup. |

Calendar Navigation

Use the following list of keys to navigate the currently focused Calendar after the popup has opened.

| Press | To do this |

| --- | --- |

| Upper Arrow | Focuses the same day of the previous week. |

| Down Arrow | Focuses the same day of the next week. |

| Left Arrow | Focuses the day before. |

| Right Arrow | Focuses the next day. |

| Home | Focuses the first day of the month. |

| End | Focuses the last day of the month. |

| Page Up | Focuses the same date of the previous month. |

| Page Down | Focuses the same date of the next month. |

| Enter | Selects the currently focused date. |

| Shift + Page Up | Focuses the same date for the previous year. |

| Shift + Page Down | Focuses the same date for the next year. |

| Control + Home | Focuses the first date of the current year. |

| Control + End | Focuses the last date of the current year. |

| Alt + Right | Focuses through out the pop-up container in forward direction. |

| Alt + Left | Focuses through out the pop-up container in backward direction. |

To focus the DateRangePicker component, use the **alt+t** keys.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, HostListener, ViewChild } from '@angular/core';
import { CalendarComponent, DateRangePickerComponent } from '@syncfusion/ej2-angular-calendars';
@Component({
  imports: [

    FormsModule,
    DateRangePickerModule
```

```

    ],
    standalone: true,
    selector: 'app-root',
    template: `
      <ejs-daterangepicker #ejDateRangePicker placeholder='Select a
range'></ejs-daterangepicker>
    `
  })
  export class AppComponent {
    @ViewChild('ejDateRangePicker') ejDateRangePicker?: CalendarComponent;
    @HostListener('document:keyup', ['$event'])
    handleKeyboardEvent(event: KeyboardEvent) {
      if (event.altKey && event.keyCode === 84 /* t */) {
        // press alt+t to focus the control.
        (this.ejDateRangePicker as any).inputElement.focus();
      }
    }
    constructor() {
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Ensuring accessibility

The DateRangePicker component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the DateRangePicker component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the DateRangePicker component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Style appearance in Angular Daterangepicker component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of DateRangePicker wrapper element

Use the following CSS to customize the appearance like height and font size of the wrapper element.

`css

/ To specify height and font size /

```

.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input {
font-size: 20px;

```

```
height: 40px;
```

```
}
```

```
,
```

Customizing the DateRangePicker icon element

Use the following CSS to customize the DateRangePicker icon element

```
`css
```

```
/ To specify background color and font size /
```

```
.e-input-group .e-input-group-icon:last-child, .e-input-group.e-control-wrapper .e-input-group-icon:last-child {
```

```
background-color: darkgray;
```

```
font-size: 14px;
```

```
}
```

```
,
```

Customizing the DateRangePicker popup calendar header

Use the following CSS to customize the DateRangePicker popup calendar header

```
`css
```

```
/ To specify background and height /
```

```
.e-daterangepicker.e-popup .e-range-header {
```

```
background: beige;
```

```
height: 80px;
```

```
}
```

```
,
```

Customizing the DateRangePicker popup calendar header title

Use the following CSS to customize the DateRangePicker popup calendar header title

```
`css
```

```
/ To specify color and font size /
```

```
.e-daterangepicker.e-popup .e-range-header .e-start-label, .e-daterangepicker.e-popup .e-range-header .e-end-label {
```

```
color: brown;
```

```
font-size: 30px;
```

```
}
```

```
,
```

Customizing the DateRangePicker popup calendar content

Use the following CSS to customize the DateRangePicker popup calendar content

```
`css
```


/ To specify background color /

```
.e-daterangepicker.e-popup .e-calendar {  
background-color: brown;  
}  
,
```

Customizing the DateRangePicker popup calendar content title

Use the following CSS to customize the DateRangePicker popup calendar content title

```
`css
```

/ To specify color and font size /

```
.e-daterangepicker.e-popup .e-calendar .e-header .e-title {  
color: beige;  
font-size: 20px;  
}  
,
```

Customizing the DateRangePicker popup calendar previous and next icon

Use the following CSS to customize the DateRangePicker popup calendar previous and next icon

```
`css
```

/ To specify font size /

```
.e-calendar .e-header .e-prev, .e-calendar .e-header .e-next, .e-bigger.e-small .e-calendar .e-header .e-  
prev, .e-bigger.e-small .e-calendar .e-header .e-next {  
font-size: 20px;  
}  
,
```

Customizing the DateRangePicker popup calendar date cell grid on hovering

Use the following CSS to customize the DateRangePicker popup calendar date cell grid on hovering

```
`css
```

/ To specify background color and border /

```
.e-calendar .e-content td:hover span.e-day {  
background-color: beige;  
border: 1px solid black;  
}  
,
```

Customizing the DateRangePicker popup calendar primary button in footer

Use the following CSS to customize the DateRangePicker popup calendar primary button in footer

```
`css
/ To specify background color and border color /

.e-daterangepicker.e-popup .e-footer .e-btn.e-apply.e-flat.e-primary:disabled, .e-daterangepicker.e-
popup .e-footer .e-btn.e-apply.e-flat.e-primary:disabled, .e-daterangepicker.e-popup .e-footer .e-css.e-
btn.e-apply.e-flat.e-primary:disabled, .e-daterangepicker.e-popup .e-footer .e-css.e-btn.e-apply.e-flat.e-
primary:disabled {
background-color: brown;
border-color: black;
}
`
```

[Customizing the DateRangePicker popup calendar cancel button in footer](#)

Use the following CSS to customize the DateRangePicker popup calendar cancel button in footer

```
`css
/ To specify background color, color, and border color /

.e-daterangepicker.e-popup .e-footer .e-btn.e-flat, .e-daterangepicker.e-popup .e-footer .e-css.e-btn.e-
flat {
background-color: beige;
border-color: black;
color: maroon;
}
`
```

[Customizing the footer element in the DateRangePicker popup calendar](#)

Use the following CSS to customize the DateRangePicker popup calendar footer element

```
`css
/ To specify background color, color, and border color /

.e-daterangepicker.e-popup .e-footer {
background-color: beige;
height: 50px;
}
`
```

[Customizing the selected date cell grid in the DateRangePicker popup calendar](#)

Use the following CSS to customize the selected date cell grid in the DateRangePicker popup calendar

```
`css
/ To specify background and border /

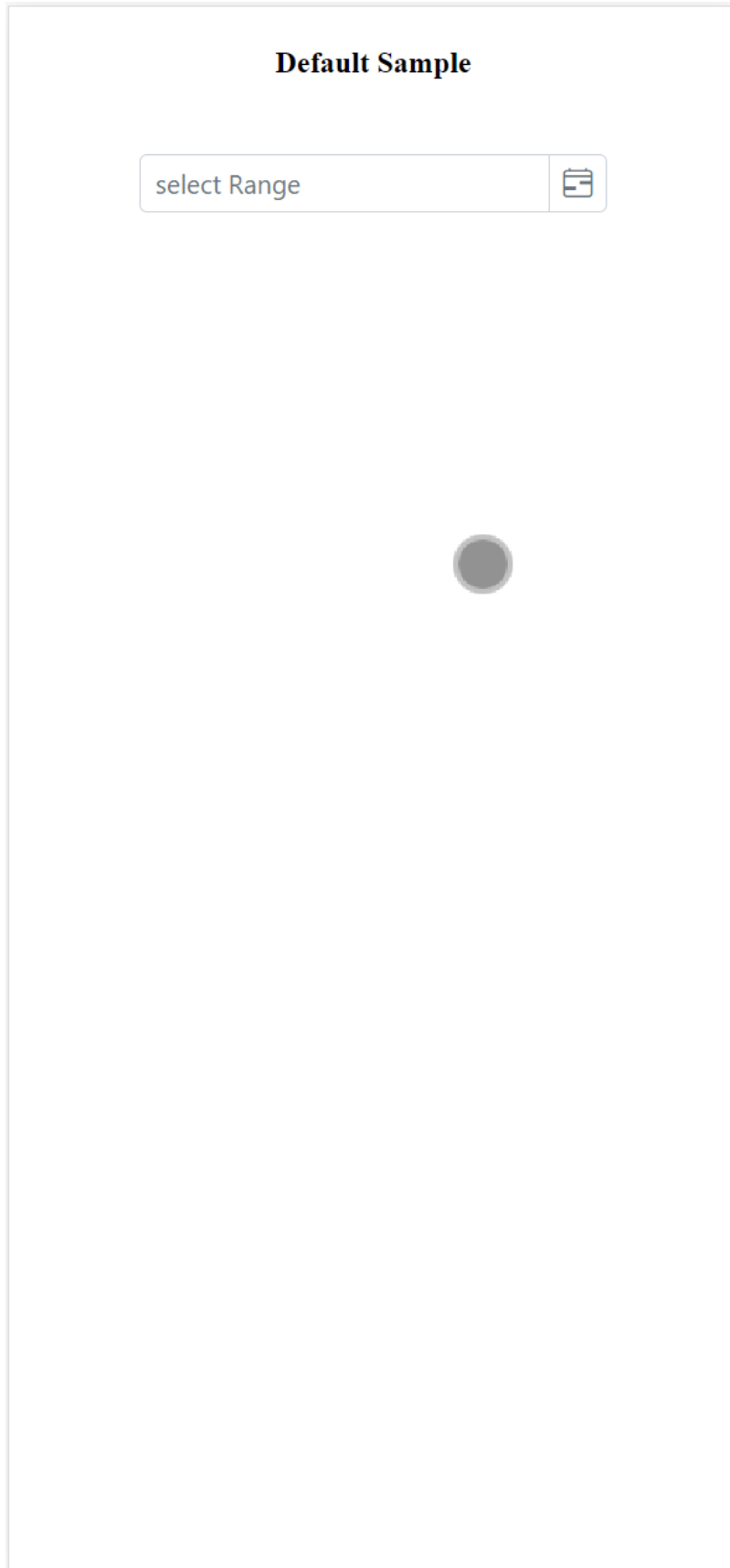
.e-calendar .e-content td.e-focused-date.e-today span.e-day {
```

```
background: lightgrey;  
border: 1px solid black;  
}  
,
```

Full screen mode support in mobiles and tablets

The DateRangePicker component's full-screen mode feature enables users to view the component popup element in full-screen mode on mobile devices with improved visibility and a better user experience. It is important to mention that this feature is exclusively available for mobile and tablet devices in both landscape and portrait orientations. To activate the full screen mode within the DateRangePicker component, simply set the [fullScreenMode](#) API value to `true`. This action will extend the calendar and presets popup element to occupy the entire screen on mobile devices.

```
`javascript  
import { Component } from '@angular/core';  
  
@Component({  
  selector: 'app-root',  
  template: `<!-- To Render daterangepicker -->  
  <ejs-daterangepicker [fullScreenMode]="true"></ejs-daterangepicker>`  
})  
  
export class AppComponent { }  
,
```



![DateRangePickerPresetsFullScreen](../images/DateRangePickerrPresetsFullScreen.gif)

How To

Two way binding in Angular Daterangepicker component

The following example demonstrates how to achieve **two-way binding** by binding the **value** to the first DateRangePicker component by using property binding and binding the model data using **ngModel** to the second DateRangePicker component. The **value** of the DateRangePicker will get change, when their is any change in the property value or model value.

The two-way binding can also be achieved only by using **property binding** or **model binding** in the DateRangePicker component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
import { DateRangePickerComponent } from '@syncfusion/ej2-angular-calendars';
@Component({
  imports: [

    FormsModule,
    DateRangePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <!-- two-way binding using the value binding and model binding in the
    DateRangePicker -->
    <ejs-daterangepicker #ejDateRangePicker [(value)]='value'
    width="230px"></ejs-daterangepicker>
    <ejs-daterangepicker #ejDateRangePickers [(ngModel)]='value'
    width="230px"></ejs-daterangepicker>
  `
})
export class AppComponent {
  value: Date;
  constructor() {
    this.value = [new Date('1/1/2020'), new Date('2/1/2023')] as any;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Disable placeholder readonly in Angular Daterangepicker component

Property | Purpose

[enabled](#) | The component can be restricted on a page, by setting `enabled` value as **false** which will disable the component completely from all user interactions including in form post action.

[placeholder](#) | Using `placeholder` you can display a short hint about the expected value in the input element.

[readonly](#) | Editing the value in the component can be prevented by setting `readonly` as **true**, but value can be included in the form post action.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule,
    DateRangePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './template.html'
})
export class AppComponent {
  public value: Date[] = [new Date('1/1/2020'), new Date('2/1/2023')];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customization using cssclass in Angular Daterangepicker component

To customize UI, you can make use of [cssClass](#) which will be added to DateRangePicker component as the root CSS class.

With this CSS class, you can override existing styles of DateRangePicker.

Following is the list of classes that provides flexible way to customize the DateRangePicker component.

Class Name	Description
---	---
e-date-range-wrapper	Applied to DateRangePicker wrapper
e-range-icon	Applied to the DateRangePicker icon.
e-popup	Applied to DateRangePicker popup wrapper.
e-calendar	Applied to both Calendar element.
e-right-calendar	Applied to right Calendar element.

- | e-left-calendar | Applied to left Calendar element. |
- | e-start-label | Applied to start label in popup. |
- | e-end-calendar | Applied to end label in a popup. |
- | e-day-span | Applied to day span details label in a popup. |
- | e-footer | Applied to footer elements in a popup. |
- | e-apply | Applied to apply button in footer in a popup. |
- | e-cancel | Applied to cancel button in footer in a popup. |
- | e-header | Applied to Calendar header. |
- | e-title | Applied to Calendar title. |
- | e-icon-container | Applied to Calendar previous and next icon container. |
- | e-prev | Applied to Calendar previous icon. |
- | e-next | Applied to Calendar next icon. |
- | e-weekend | Applied to Calendar weekend dates. |
- | e-other-month | Applied to Calendar other month dates. |
- | e-day | Applied to each day cell of the Calendar. |
- | e-selected | Applied to Calendar selected dates. |
- | e-disabled | Applied to Calendar disabled dates. |

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
@Component({
  imports: [
    DateRangePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./style.css'],
  template: `<ejs-daterangepicker #ejDateRangePicker cssClass='customCSS'
placeholder='Select a range'></ejs-daterangepicker>`
})
export class AppComponent {
  constructor() {}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Custom event emitter in Angular Daterangepicker component

The **two-way binding** in DateRangePicker can also be achieved using the custom event binding and property binding in the controls present in two different components. To create custom event, we need to create an instance of **event emitter**.

In the following example, **property binding** is used to share the data from the parent component to the child component using **@input** directive and **custom event binding** is used to share the data from the child component to the parent component by using **@output** directive.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
import { DateRangePickerComponent } from '@syncfusion/ej2-angular-calendars';
@Component({
  imports: [ DateRangePickerModule ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div class="parentelement">
      <div class="rangevalue">
        <ejs-daterangepicker id="daterangepicker" #range (change)="deposit()"
        [value]="value" placeholder="Parent component" floatLabelType="Always"
        width="200px"></ejs-daterangepicker>
      </div>
    </div>
    <child [xvalue]="value" (valueChange)="valuecheck($event)"> </child>
  `
})
export class ParentComponent {
  @ViewChild('range')
  public DateRange?: DateRangePickerComponent;
  value: Date[];
  constructor() {
    this.value = [new Date("1/1/2019"), new Date("2/1/2020")]
  }
  deposit() {
    this.value = (this.DateRange as DateRangePickerComponent).value as
    Date[];
  }
  valuecheck(args: any) {
    this.value = args;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```


Custom validation using form validator in Angular Daterangepicker component

The client side validation takes place in the browser to avoid the waiting time to receive the response from sever. It validates the form elements to provide the better feedback messages to correct the every fields before the form submission.

The following sample demonstrate how to achieve the client side validation in DateRangePicker component by using **FormValidator** function. It provides an option to customize the feedback error messages to the corresponding fields to take action to resolve the issue.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild, OnInit } from '@angular/core';
import { DateRangePickerComponent } from '@syncfusion/ej2-angular-calendars';
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
@Component({
  imports: [
    DateRangePickerModule,
    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<form id="form-element" class="form-vertical">
    <ejs-daterangepicker #ejDateRange id='daterangepicker' placeholder='Enter
a date range' width="275px"(blur)="onFocusOut()" (change)=
"onChange($event)"></ejs-daterangepicker>
  </form>`
})
export class AppComponent implements OnInit {
  @ViewChild('formElement') element: any;
  @ViewChild('ejDateRange') ejDateRange?: DateRangePickerComponent;
  public formObject?: FormValidator;
  ngOnInit() {
    let customFn: (args: {
      [key: string]: string
    }) => boolean = (args: {
      [key: string]: string
    }) => {
      return ((this.ejDateRange as any).value[0]).getFullYear() > 1990
      && ((this.ejDateRange as DateRangePickerComponent |
      any).value[1]).getFullYear() < 2020;
    };
    let options: FormValidatorModel = {
      rules: {
        'daterangepicker': {
          required: [true, "Value is required"]
        }
      },
      customPlacement: (inputElement: HTMLElement, errorElement:
      HTMLElement) => {
        inputElement?.parentElement?.parentElement?.appendChild(errorElement);
      }
    };
  }
}
```

```

    }
    };
    this.formObject = new FormValidator('#form-element', options);
    this.formObject.addRules('daterangepicker', {
      range: [customFn, "Please select a date range between years
from 1990 to 2020"]
    });
    this.formObject = new FormValidator('#form-element', options);
  }
  public onFocusOut(): void {
    this.formObject?.validate("daterangepicker");
  }
  public onChange(args: any) {
    if (this.ejDateRange?.value != null)
      this.formObject?.validate("daterangepicker");
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Reactive form in Angular Daterangepicker component

DateRangePicker is a form component and validation is playing vital role in forms to get the valid data.

Here to showcase the DateRangePicker with form validations we have used the reactive form.

The reactive forms uses the reactive model-driven technique to handle form data between component and view, due to that we also call it as the model-driven forms.

It's listen the form data changes between App component and view, also returns the valid states and values of form elements.

For more details about Reactive Forms refer: <https://angular.io/guide/reactive-forms>.

For the reactive forms, import a **ReactiveFormsModule** into app module as well as the **FormGroup**, **FormControl** should be imported to app component.

The **FormGroup** is used to declare **formGroupName** for the form and the **FormControl** is used to declare **formControlName** for form controls. Declare the **formControlName** to DateRangePicker component as usual.

Then, create a value object to the **FormGroup** and each value will be the default value of the form control.

The following example demonstrates how to use the reactive forms.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars';

```

```

import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit, Inject } from '@angular/core';
import { DateRangePickerComponent } from '@syncfusion/ej2-angular-calendars';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
import { FormsModule } from '@angular/forms';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, DateRangePickerModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './template.html'
})
export class AppComponent implements OnInit {
  skillForm?: FormGroup | any;
  build: FormBuilder;
  constructor(@Inject(FormBuilder) private builder: FormBuilder) {
    this.build = builder;
    this.createForm();
  }
  ngOnInit(): void {
    throw new Error('Method not implemented.');
```

```

  }
  createForm() {
    this.skillForm = this.build.group({
      daterangepicker: ['', Validators.required],
      username: ['', Validators.required],
      usermail: ['', Validators.email],
    });
  }
  get username() {
    return this.skillForm?.get('username');
  }
  get daterangepicker() {
    return this.skillForm?.get('daterangepicker');
  }
  get usermail() {
    return this.skillForm?.get('usermail');
  }
  onSubmit() {
    alert("Form Submitted!");
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Template driven forms in Angular Daterangepicker component

The form can be build with Angular template syntax easily along with form specific directives.

This template-driven forms uses the `ng` directives in view to handle the forms controls.

- To enable the template-driven, import the FormsModule into corresponding app component.

For more details about template-driven Forms refer to: <https://angular.io/guide/forms#template-driven-forms>.

- In angular forms mentioning the name is must to process as form elements.
- Mention the `name` attribute to DateRangePicker element which will be used to identify the form element. To register an DateRangePicker element to ngForm, give the ngModel to it so the FormsModule will automatically detect the DateRangePicker as a form element.

After that, the DateRangePicker value will be selected based on the ngModel value.

The following example demonstrates template driven forms with DateRangePicker component.

APP.COMPONENT.TS

```
import { BrowserModule } from '@angular/platform-browser'
import { NgModule } from '@angular/core'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild, ViewEncapsulation, Inject } from '@angular/core';
import { FormGroup, FormBuilder, Validators } from '@angular/forms';
import { DateRangePickerComponent } from '@syncfusion/ej2-angular-calendars';

class User {
  public range?: Date[];
  constructor() {
  }
}

@Component({
  imports: [ FormsModule, ReactiveFormsModule, DateRangePickerModule],
  standalone: true,
  selector: 'app-root',
  templateUrl: './template.html'
})
export class DefaultDateRangePickerComponent {
  user?: User | any;
  ngOnInit() {
    this.user = new User();
  }
  onSubmit(userForm: any) {
    (userForm.valid) ? alert("submitted") : alert("form is invalid");
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

TEMPLATE.HTML

```

<div>
  <form #userForm="ngForm" (ngSubmit)="onSubmit(userForm)">
    <div class="form-group">
      <ejs-daterangepicker id="daterangepicker" name="date"
[ (ngModel) ]="user.date" #date="ngModel" width="350px" required></ejs-
daterangepicker>
      <div *ngIf="userForm.invalid && (userForm.dirty ||
userForm.touched)" class="alert alert-danger formerror">
        Valid date is required
      </div>
      <div *ngIf="userForm.valid && (userForm.dirty || userForm.touched)"
class="alert alert-success formerror">
        <table>
          <tr>
            <td style="width:50%">Selected value: </td>
            <td class="formtext ">{{ user.date[0] | date:"dd/MM/yyyy"}} -
{{ user.date[1] | date:"dd/MM/yyyy"}} </td>
          </tr>
        </table>
      </div>
    </div>
    <div class="buttons">
      <button type="submit" class="e-btn e-success">Submit</button>
      <button type="reset" class="e-btn e-warning">Reset</button>
    </div>
  </form>
  <div class="dataclass">userForm.value: {{userForm.value | json}}</div>
  <div class="dataclass">userForm.valid: {{userForm.valid}}</div>
</div>
<style>
  form {
    margin-top: 50px;
  }
</style>

```

Customize the daterangepicker day header in Angular Daterangepicker component

You can change the format of the day that to be displayed in header using [dayHeaderFormat](#) property.

By default, the format is Short.

You can find the possible formats on below.

| **Name** | **Description** |

|-----|-----|

| **Short** | Sets the short format of day name (like Su) in day header. |

| **Narrow** | Sets the single character of day name (like S) in day header. |

| **Abbreviated** | Sets the min format of day name (like Sun) in day header. |

| **Wide** | Sets the long format of day name (like Sunday) in day header. |

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
```

```

import { BrowserModule } from '@angular/platform-browser'
import { DateRangePickerModule } from '@syncfusion/ej2-angular-calendars'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, ViewChild } from '@angular/core';
import { DateRangePickerComponent } from '@syncfusion/ej2-angular-calendars';
import { DropDownListComponent, ChangeEventArgs } from '@syncfusion/ej2-
angular-dropdowns';
@Component({
  imports: [

    DropDownListModule,
    DateRangePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./style.css'],
  template: `
    <div id="container">
      <div id="daterangepicker">
        <ejs-daterangepicker #default cssClass="format-wide"
dayHeaderFormat='Short'></ejs-daterangepicker>
      </div>
      <div id="format">
        <label class="custom-input-label">Header Format Types</label>
        <div id="wrap">
          <ejs-dropdownlist id="dayformat" #select
[dataSource]='formatData' [(value)]= 'value' [fields]='fields'
[placeholder]='waterMark' (change)='formatHandler($event)'></ejs-
dropdownlist>
        </div>
      </div>
    </div>
  `
})
export class AppComponent {
  @ViewChild('default')
  public daterangepickerObj?: DateRangePickerComponent;
  @ViewChild('select')
  public dayHeaderFormat?: DropDownListComponent;
  // define the JSON of data
  public formatData: Object[] = [
    { Id: 'Short', Label: 'Short' },
    { Id: 'Narrow', Label: 'Narrow' },
    { Id: 'Abbreviated', Label: 'Abbreviated' },
    { Id: 'Wide', Label: 'Wide' },
  ];
  public fields: Object = { text: 'Label', value: 'Id' };
  public waterMark: string = 'Select format type';
  public value: string = 'Short';
  public formatHandler(args: ChangeEventArgs): void {
    (this.daterangepickerObj as DateRangePickerComponent |
any).dayHeaderFormat = args.value;
  }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Ej1 api migration in Angular Daterangepicker component

This article describes the API migration process of DateRangePicker component from Essential JS 1 to Essential JS 2.

Date Selection

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Setting the value	Property: value	Property: valuepublic value: Date[] = [new Date("10/07/2017"), new Date("2/2/2019")]

Date Format

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Display date format	Property: dateFormat	Property: format

Date Range

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Minimum date	Property: minDate	Property: min
Maximum date	Property: maxDate	Property: max
Start date	Property: startDate	Property: startDate
End date	Property: endDate	Property: endDate
Preset ranges	Property: ranges ranges: Object = [{ label: "Today", range: [new Date(), new Date()] }, { label: "Last Week", range: [new Date(new Date().setDate(new Date().getDate() - 7)), new Date()] }]	Property: presets public today: Date = new Date(new Date().toDateString()); public weekStart: Date = new Date(new Date(new Date().setDate(new Date().getDate() - (new Date().getDay() + 7) % 7)).toDateString()); public weekEnd: Date = new Date(new Date(new Date().setDate(new Date(new Date().setDate((new Date().getDate() - (new Date().getDay() + 7) % 7))).getDate() + 6)).toDateString());
Add ranges	Method: addRanges() public onCreate(e:any) { var dateObj = \$("#daterangepicker").data("ejDateRangePicker"); dateObj.addRanges("new Range", [new Date("11/12/2019"), new Date("11/12/2021")]); }	Not Applicable
Clear ranges	Method: clearRanges() public onCreate(e:any) { var dateObj = \$("#daterangepicker").data("ejDateRangePicker"); dateObj.clearRanges(); }	Not Applicable
Get selected range	Method: getSelectedRange() public onCreate(e:any) { var dateObj = \$("#daterangepicker").data("ejDateRangePicker"); console.log(dateObj.getSelectedRange()); }	Method: getSelectedRange()@ViewChild("rangeObj") rangeObj: DateRangePickerComponent; public onCreate (args: any) { console.log(this.rangeObj.getSelectedRange()); }

Set date range	Method: setRange() public onCreate (args: any) { var dateObj = \$("#daterangepicker").data("ejDateRangePicker"); dateObj.setRange([new Date("11/12/2011"), new Date("11/12/2019")]);}	Not Applicable
----------------	--	----------------

Disabled Dates

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Disabled dates	Not Applicable	Can be achieved by public disableDaterange(args:any) { if (args.date.getDay() === 0 args.date.getDay() === 6) { args.isDisabled = true; }}

Customization

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
CSS Class	Property: cssClass	Property: cssClass
Enable/Disable the TimePicker with DateRangePicker	Property: enableTimePicker	Not Applicable
Time format	Property: timeFormat	Not Applicable
Minimum days span	Not Applicable	Property: minDays
Maximum days span	Not Applicable	Property: maxDays
Button text	Property: buttonText buttonText: Object = { reset: "Again", cancel: "Cut", apply: "Ok" }	Can be achieved by <code>L10n.load({ 'en': { 'daterangepicker': { applyText: 'Apply' } } });</code>
Show/Hide the popup button	Property: showPopupButton	Event: focus <code>@ViewChild("rangeObj") rangeObj: DateRangePickerComponent; public onFocus(args:any) { this.rangeObj.show(); } css.e-control-wrapper .e-input-group-icon.e-range-icon { display: none; }</code>
Enable/Disable the rounded corner	Property: showRoundedCorner	Can be achieved by <code>css.e-control-wrapper.e-custom-style.e-date-range-wrapper.e-input-group { border-radius: 4px; }</code>
FocusIn event	Not Applicable	Event: focus <code>public onFocus(args:any) { /* code block */ }</code>
FocusOut event	Not Applicable	Event: blur <code>public onBlur(args:any) { /* code block */ }</code>
FocusIn method	Not Applicable	Method: focusIn() <code>@ViewChild("rangeObj") rangeObj: DateRangePickerComponent; public onCreate(args:any) { this.rangeObj.focusIn(); }</code>

FocusOut method	Not Applicable	Method: focusOut()@ViewChild("rangeObj") rangeObj: DateRangePickerComponent;public onCreate(args:any) { this.rangeObj.focusOut();}
-----------------	----------------	---

Accessibility

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Enable/Disable the RTL	Not Applicable	Property: enableRtl

Persistence

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Enable/Disable the persistence	Property: enablePersistence	Property: enablePersistence

Common

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Width	Property: width	Property: width
Readonly	Not Applicable	Property: readonly
Show/Hide the clear button	Not Applicable	Property: showClearButton
Height	Property: height	Can be achieved bycss.e-control-wrapper.e-custom-style.e-date-range-wrapper.e-input-group { height: 35px;}
Show/Hide the week number	Not Applicable	Property: weekNumber
Watermark text	Property: watermarkText	Property: placeholder
Enable/Disable	Property: enabled	Property: enabled
Disable the DateRangePicker	Method: disable() public onCreate(e:any) { var daterangeObj = \$("#daterangepicker").data("ejDateRangePicker"); daterangeObj.disable();}	Not Applicable
Enable the DateRangePicker	Method: enable() public onCreate(e:any) { var daterangeObj = \$("#daterangepicker").data("ejDateRangePicker"); daterangeObj.enable();}	Not Applicable
Enable/Disable the input textbox editing	Property: allowEdit	Property: allowEdit
Specify the placeholder text behavior	Not Applicable	Property: floatLabelType
zIndex	Not Applicable	Property: zIndex
Separator	Property: separator	Property: separator

Globalization

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Locale	Property: locale	Property: locale
First day of week	Not Applicable	Property: firstDayOfWeek

Strict mode

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Strict mode	Not Applicable	Property: strictMode

Open and Close

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Close	Event: closepublic onClose(e:any) {/ code block */}	Event: closepublic onClose(args:any) {/ code block */}
Hide	Method: popupHide()public onCreate(e:any) { var daterangeObject = \$("#daterangepicker").data("ejDateRangePicker"); daterangeObject.popupShow(); daterangeObject.popupHide();}	Method: hide()@ViewChild("rangeObj") rangeObj: DateRangePickerComponent;public create(args:any) { this.rangeObj.show(); this.rangeObj.hide();}
Open	Event: openpublic onOpen(e:any) {/ code block */}	Event: openpublic onOpen(args:any) {/ code block */}
Show	Method: popupShow()public onCreate(e:any) { var daterangeObject = \$("#daterangepicker").data("ejDateRangePicker"); daterangeObject.popupShow();}	Method: show()@ViewChild("rangeObj") rangeObj: DateRangePickerComponent;public create(args:any) { this.rangeObj.show();}