

USER GUIDE

Essential Studio

for EJ2 Angular

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

Toolbar	18
Getting started with Angular Toolbar component	18
Dependencies.....	18
Setup Angular Environment.....	18
Create an Angular Application	18
Installing Syncfusion Toolbar Package	18
Registering Toolbar Module.....	19
Adding CSS reference.....	20
Add Toolbar component	20
Initialize the Toolbar using HTML elements	21
See Also	22
Item configuration in Angular Toolbar component	23
Button	23
Separator.....	23
Input.....	24
See Also	29
Template configuration in Angular Toolbar component	29
Integrate menu component.....	29
Responsive mode in Angular Toolbar component.....	31
Scrollable.....	31
Popup.....	33
Render toolbar with a less than minimum height	35
Accessibility in Angular Toolbar component	37
ARIA attributes.....	38
Keyboard interaction	38
Ensuring accessibility	39
See also	39
Style in Angular Toolbar component	39
Customizing Toolbar	39
Customizing the Toolbar items	39
Customizing Toolbar's item icon	39
Customizing the hover state of Toolbar control.....	40
Customizing selected item of Toolbar control.....	40
How To	40
Set command customization in Angular Toolbar component	40

Set tool tip to the commands in Angular Toolbar component	41
Set item wise custom template in Angular Toolbar component	42
Add toggle button in Angular Toolbar component.....	43
Add font awesome in Angular Toolbar component.....	46
How to customize toolbar scroll step in Angular Toolbar component	47
Add link to toolbar item in Angular Toolbar component.....	49
Render other components in toolbar using angular template in Angular Toolbar component.....	50
Ej1 api migration in Angular Toolbar component.....	51
Accessibility and Localization.....	51
DataSource.....	52
Items	52
Common.....	55
Tooltip	57
Getting started with Angular Tooltip component	57
Dependencies.....	57
Setup Angular Environment.....	57
Create an Angular Application	58
Installing Syncfusion Tooltip package	58
Registering Tooltip Module.....	59
Adding CSS Reference	59
Add Tooltip component	59
Run the application	60
See Also	62
Schematics in Angular Tooltip component	62
Getting started	62
Dependency and Module injection using Schematics	63
Component generation using Schematics	63
Setting dimension in Angular Tooltip component	64
Height and width.....	64
Content in Angular Tooltip component	66
Template content.....	66
Dynamic content via AJAX.....	67
Position in Angular Tooltip component	69
Tip pointer positioning.....	71
Dynamic positioning.....	72

Mouse trailing	74
Setting offset values	75
Open mode in Angular Tooltip component	76
Custom open mode	78
Sticky mode	79
Open/Close Tooltip with delay	80
Animation in Angular Tooltip component	81
Animation effects	82
Animating via open/close methods	83
Apply transition	84
Customization in Angular Tooltip component	86
Tip pointer customization	86
Tooltip customization	88
Style in Angular Tooltip component	89
Customizing the tooltip	89
Customizing the tooltip popup	89
Customizing the tooltip content	89
Customizing the tooltip arrow tip	89
Customizing the tooltip inner tip	90
Customizing the tooltip outer tip	91
Accessibility in Angular Tooltip component	92
WAI-ARIA attributes	93
Keyboard interaction	93
Ensuring accessibility	93
See also	94
Ej1 api migration in Angular Tooltip component	94
How To	95
Show tooltip on disabled elements and disable tooltip in Angular Tooltip component	95
Dynamic tooltip content with html elements in Angular Tooltip component	96
Custom tooltip with dynamic html in Angular Tooltip component	97
Tooltip open or display modes in Angular Tooltip component	98
Fancy tooltip customization in Angular Tooltip component	100
Display tooltip on svg and canvas elements in Angular Tooltip component	103
Create and show tooltip on multiple targets in Angular Tooltip component	105
Dynamic tooltip content in Angular Tooltip component	107

Tooltip content template in Angular Tooltip component.....	109
TreeGrid	110
Getting started with Angular Treegrid component	110
Setup Angular Environment.....	110
Create an Angular Application	110
Installing Syncfusion TreeGrid package	111
Registering TreeGrid Module.....	111
Adding CSS reference.....	112
Add TreeGrid component	112
Defining Row Data	113
Defining Columns	113
Module injection.....	115
Enable Paging.....	115
Enable Sorting.....	116
Enable Filtering	117
Run the application	118
Modules in Angular Treegrid component.....	120
Data Binding.....	120
Data binding in Angular Treegrid component	120
Local data in Angular Treegrid component.....	125
Remote data in Angular Treegrid component	130
Immutable mode in Angular Treegrid component	159
Limitations.....	161
Observables in Angular Treegrid component	162
Observable binding using Async pipe	162
Data binding.....	162
Handling child data	164
Handling TreeGrid actions	167
Perform CRUD operations.....	168
Calculate aggregates	172
Provide Excel Filter data source	172
Columns in Angular TreeGrid component	175
Column types	175
Column width.....	177
Column formatting.....	179

Align the text of content	185
Render boolean values as checkbox	187
AutoFit columns	190
Locked columns.....	194
Show or hide columns.....	196
Controlling Tree Grid actions	200
Customize column styles.....	201
Manipulating columns	201
Responsive columns.....	205
Row in Angular TreeGrid component	206
Customize row styles	207
Styling alternate rows	211
Styling parent and child rows.....	212
Row height	214
Row hover	217
Adding a new row programmatically.....	222
Show or hide a row using an external actions	225
How to get the row data and element.....	227
See also	228
Cell	228
Sorting in Angular TreeGrid component.....	228
Initial sorting	230
Multi-column sorting	231
Prevent sorting for particular column.....	233
Sort order	234
Custom sorting.....	235
Touch interaction	236
Perform sorting based on its culture	236
Customize sort icon.....	238
Sort columns externally	240
Sorting events	245
Filtering	247
Filtering in Angular TreeGrid component	247
Filter bar in Angular TreeGrid component.....	310
Filter menu in Angular TreeGrid component.....	342

Excel like filter in Angular TreeGrid component.....	387
Searching in Angular TreeGrid component.....	426
Initial search.....	431
Search by external button.....	436
Search specific columns	442
Search on each key stroke	447
Perform search based on column formatting.....	452
Perform search operation in Grid using multiple keywords.....	458
How to ignore accent while searching.....	464
Highlight the search text.....	468
Clear search by external button.....	474
See also	479
Paging in Angular TreeGrid component.....	479
Customize the pager options	480
Page size Mode	502
Pager template.....	507
Pager with page size dropdown.....	512
How to navigate to particular page	523
How to get the pager element.....	528
Dynamically calculate page size based on element height.....	528
Render pager at the top of the tree grid	534
Pager events.....	539
See Also	544
Scrolling in Angular Treegrid component	545
Set width and height.....	545
Responsive with parent container	546
Scroll to selected row.....	547
Hide the scrollbar when the content is not overflown	548
Frozen in Angular Treegrid component	549
Frozen rows and columns	549
Add validation rule for frozen tree grid	554
Virtual scroll in Angular Treegrid component.....	556
Row virtualization	556
Column virtualization.....	557
Limitations for virtualization.....	559

Infinite scroll in Angular Treegrid component	560
InitialBlocks	561
Cache Mode	562
Limitations for Virtualization	563
Selection.....	564
Selection in Angular Treegrid component	564
Cell selection in Angular Treegrid component.....	568
Check box selection in Angular Treegrid component	572
Row selection in Angular Treegrid component	574
Aggregates	581
Aggregates in Angular Treegrid component	581
Footer aggregate in Angular Treegrid component	582
Custom aggregate in Angular Treegrid component.....	585
Tool Bar	586
Tool bar in Angular Treegrid component.....	586
Tool bar items in Angular Treegrid component.....	589
Custom tool bar in Angular Treegrid component	592
Edit in Angular TreeGrid component	595
Toolbar with edit option	600
Disable editing for particular column	605
Disable editing for particular row	611
Editing template column.....	616
Customize delete confirmation dialog.....	621
Update boolean column value with a single click.....	627
Edit enum column	632
Edit complex column.....	635
How to perform CRUD action externally	638
Troubleshoot editing works only for first row	650
How to make a tree grid column always editable	650
See also	655
Print in Angular Treegrid component	655
Page setup.....	656
Print using an external button	656
Print the visible page.....	658
Print large number of columns	659

Show or Hide columns while Printing	659
Limitations of Printing Large Data.....	661
Adaptive in Angular Treegrid component.....	661
Render adaptive dialogs.....	661
Loading animation in Angular Treegrid component.....	663
State Persistence.....	664
State persistence in Angular Treegrid component	664
Get or set local storage value in Angular Treegrid component	664
Pdf Export.....	665
Pdf export in Angular Treegrid component	665
Pdf export options in Angular Treegrid component	670
Pdf cell style customization in Angular Treegrid component	683
Adding header and footer in Angular Treegrid component	686
Exporting tree grid in server in Angular Tree Grid component	691
Excel Export.....	695
Excel export in Angular Treegrid component	695
Excel export options in Angular Treegrid component	699
Excel cell style customization in Angular Treegrid component	703
Adding header and footer in Angular Treegrid component	707
Exporting tree grid in server in Angular Tree Grid component	708
Context menu in Angular Treegrid component.....	714
Custom context menu items.....	716
Enable and disable context menu items dynamically.....	718
Accessibility in Angular Treegrid component	720
WAI-ARIA attributes.....	721
Keyboard interaction	722
Ensuring accessibility	723
See also	723
Clipboard in Angular Treegrid component	723
Copy to clipboard by external buttons	725
AutoFill	728
Paste.....	729
Global local in Angular Treegrid component	731
Localization	731
Internationalization.....	736

Right to left (RTL)	738
See Also	740
Treegrid styling in Angular Treegrid component	740
How To	741
Refresh the data source in Angular Treegrid component	741
Enable disable treegrid and its actions in Angular Treegrid component.....	744
Change header text dynamically in Angular Treegrid component	746
Customize column styles in Angular Treegrid component	748
Custom tool tip for columns in Angular Treegrid component	750
Change orientation of header text in Angular Treegrid component	752
Render other component in column in Angular Treegrid component	754
Customize the icon for column menu in Angular Treegrid component	755
Customize the edit dialog in Angular Treegrid component.....	757
Cascading drop down list with treegrid editing in Angular Treegrid component.....	758
Provide custom data source and enabling filtering to drop down list in Angular Treegrid component	761
Restrict decimal points while treegrid editing in Angular Treegrid component	763
Exporting filtered data in Angular Treegrid component.....	765
Exporting selected data in Angular Treegrid component	767
Show spinner while exporting in Angular Treegrid component	769
Passing parameter to server exporting in Angular Treegrid component	771
Customize pager drop down in Angular Treegrid component	773
Add parameter for filtering in Angular Treegrid component	774
Select treegrid rows based on certain condition in Angular Treegrid component.....	775
Get row cell index in Angular Treegrid component.....	777
Display foreign key values in treegrid in Angular Treegrid component	778
Row cell customization in Angular Treegrid component.....	780
TreeMap.....	782
Getting started with Angular Treemap component	782
Setup Angular Environment.....	782
Create an Angular Application	782
Adding Syncfusion TreeMap package	782
Registering TreeMap Module	783
Render TreeMap	784
Module Injection.....	785

Apply Color Mapping	786
Enable Legend	788
Add Labels	789
Enable Tooltip	791
Data binding in Angular Treemap component	793
Populate data	794
Layout in Angular Treemap component	795
Types of layout	795
Leaf item in Angular Treemap component	798
Leaf label	798
Item gap	801
Levels in Angular Treemap component	802
Group path	802
Group gap	804
Header format and Alignment	805
Header height and style	807
Header template and position	808
Color mapping in Angular Treemap component	810
Range color mapping	810
Equal color mapping	811
Desaturation color mapping	812
Palette color mapping	813
Desaturation with multiple colors	814
Color for items excluded from color mapping	815
Bind the colors to the items from data source	816
Data label in Angular Treemap component	817
Format	817
Template	818
InterSectAction	819
Legend in Angular Treemap component	819
Position and alignment	820
Legend mode	822
Legend size	824
Legend for items excluded from color mapping	827
Hide desired legend items	828

Hide legend items based data source value	829
Bind legend item text from data source	830
Hide duplicate legend items	831
Legend Responsiveness	832
Drilldown in Angular Treemap component	833
Perform drill-down action.....	833
On-demand data loading	835
Breadcrumb support.....	836
Tooltip in Angular Treemap component.....	837
Default tooltip.....	838
Format tooltip.....	838
Tooltip template	839
Selection and highlight in Angular Treemap component	840
Selection.....	840
Highlight	842
Print and export in Angular Treemap component.....	844
Print.....	844
Export.....	845
Accessibility in Angular TreeMap component	849
WAI-ARIA attributes.....	850
Screen reading in TreeMap.....	850
Ensuring accessibility	850
See also	850
Internationalization in Angular Treemap component	850
Globalization	851
Right-to-left rendering.....	852
Legend with Rtl support.....	852
Tooltip with Rtl support	853
Treemap Item Rendering Direction	854
Ej1 api migration in Angular Treemap component.....	857
Data Binding.....	857
Appearance	857
Leaf Items.....	858
Legend.....	860
Levels.....	862

Selection.....	864
Highlight.....	864
Range ColorMapping.....	865
Desaturation ColorMapping	866
Tooltip	867
Drilldown.....	868
Methods.....	868
Events.....	869
How To	870
Drilldown in Angular Treemap component	870
Drilldown with label in Angular Treemap component.....	874
TreeView	876
Getting started with Angular Treeview component.....	876
Setup Angular Environment.....	876
Create an Angular Application	876
Installing Syncfusion Treeview Package.....	876
Registering Treeview Module	877
Adding CSS Reference	877
Add Treeview component.....	878
Binding data source	878
Run the application	879
See Also	880
Data binding in Angular Treeview component	881
Local data	881
Remote data.....	884
Check box in Angular Treeview component	886
Checked nodes	887
See Also	889
Node editing in Angular Treeview component.....	889
See Also	892
Multiple selection in Angular Treeview component.....	892
Selected nodes.....	893
See Also	895
Drag and drop in Angular Treeview component.....	895
Multiple-node drag and drop.....	897

See Also	899
Template in Angular Treeview component	899
See Also	900
Accessibility in Angular TreeView component.....	900
WAI-ARIA attributes.....	901
Keyboard interaction	901
Ensuring accessibility	902
See also	902
How To	902
Customize the expand and collapse icons in Angular Treeview component	902
Process the tree node operations using context menu in Angular Treeview component	905
Check uncheck the checkbox on clicking the tree node text in Angular Treeview component	908
Validate the text when renaming the tree node in Angular Treeview component.....	909
Customize the tree nodes based on levels in Angular Treeview component.....	911
Restrict the drag and drop for particular tree nodes in Angular Treeview component.....	914
Accordion tree in Angular Treeview component.....	916
Auto hide show expand collapse icon in Angular Treeview component.....	917
Filtering tree nodes in Angular Treeview component	919
Set tool tip for tree nodes in Angular Treeview component	921
Sorting treeview level wise in Angular Treeview component	924
Remove parent checkbox in Angular Treeview component.....	926
Get dynamic icon in Angular Treeview component.....	927
Hover multi line tree node in Angular Treeview component	929
Select one child in Angular Treeview component	931
Disable checkbox of the tree node in Angular Treeview component.....	935
Ej1 api migration in Angular Treeview component	936
Add nodes	936
Common.....	937
CheckBox.....	944
Drag and Drop.....	946
Expand/Collapse nodes.....	947
Node Editing.....	949
Node Selection	949
Template	952
Uploader	952

Getting started with Angular Uploader component.....	952
Dependencies.....	952
Setup angular environment	952
Create a new application	953
Installing Syncfusion Uploader Package	953
Initialize Uploader	954
Adding CSS reference.....	954
Adding Uploader component.....	955
Running the application.....	955
Adding drop area	956
Configure asynchronous settings.....	957
Handle success and failed upload	958
See Also	959
Async in Angular Uploader component	959
Multiple file upload.....	959
Single file upload.....	960
Save action	961
Remove action	965
Auto upload.....	967
Sequential Upload.....	968
Preloaded files	968
Adding additional HTTP headers with upload action.....	969
See Also	970
Chunk upload in Angular Uploader component	970
Additional configurations.....	971
Resumable upload	973
Cancel upload.....	973
Server-Side configurations.....	974
File source in Angular Uploader component	978
Paste to upload	978
Directory upload	980
Drag and drop	982
See Also	984
Validation in Angular Uploader component	984
File type.....	984

File size	985
Maximum files count	986
Duplicate files.....	987
See Also	988
Form support in Angular Uploader component	989
Template-driven forms	990
Reactive forms	993
Template in Angular Uploader component	996
File list template.....	996
Custom template	998
See Also	1002
Localization in Angular Uploader component	1002
Accessibility in Angular Uploader component.....	1004
Keyboard interaction	1005
Ensuring accessibility	1006
See also	1006
Style appearance in Angular Uploader component.....	1006
Customizing the appearance of File Upload wrapper element	1006
Customizing the File Upload browse button	1007
Customizing the File Upload content.....	1007
Customizing the uploaded file container in File Upload	1007
See Also	1007
How To	1008
Hide default drop area in Angular Uploader component	1008
Preview images before uploading in Angular Uploader component.....	1008
Achieve invisible upload in Angular Uploader component.....	1009
Customize progressbar in Angular Uploader component	1010
Sort the selected files in Angular Uploader component.....	1011
Get the total size of selected files in Angular Uploader component	1012
Customize button with html element in Angular Uploader component.....	1013
Add confirm dialog to remove the files in Angular Uploader component	1014
Add additional data on upload in Angular Uploader component.....	1015
Validate image on drop in Angular Uploader component.....	1016
Determine whether uploader has file input in Angular Uploader component	1017
Achieve file upload programmatically in Angular Uploader component	1019

Check file size before uploading it in Angular Uploader component	1020
Check the mime type of file before upload it in Angular Uploader component	1021
Trigger click event of input file in Angular Uploader component.....	1022
Open and edit the uploaded files in Angular Uploader component.....	1023
Resize images before upload to server.....	1025
Convert image into binary format after uploading in Angular Uploader component.....	1035
Ej1 api migration in Angular Uploader component	1036
Accessibility	1036
File List	1036
File selection	1037
Upload action	1038
Chunk Upload.....	1039
Remove action	1040
Buttons.....	1041
Drag and Drop.....	1041
Common.....	1041

Toolbar

Getting started with Angular Toolbar component

This section explains briefly about how to create a simple **Toolbar** using Angular and configuring the Toolbar items like button, separator and input components.

Dependencies

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

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

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 Toolbar 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.

Registering Toolbar Module

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

```
`javascript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// import the ToolbarModule for the Toolbar component
import { ToolbarModule } from '@syncfusion/ej2-angular-navigations';
import { AppComponent } from './app.component';

@NgModule({
  //declaration of ej2-angular-navigations module into NgModule
  imports: [ BrowserModule, ToolbarModule ],
  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-popups/styles/material.css';  
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';  
`
```

Add Toolbar component

Modify the template in `[src/app/app.component.ts]` file to render the toolbar component.

Add the Angular Toolbar by using `<ejs-toolbar>` selector in **template** section of the `app.component.ts` file.

```
`typescript  
import { Component, OnInit } from '@angular/core';  
  
@Component({  
  selector: 'app-root',  
  // specifies the template string for the Toolbar component  
  template: `<ejs-toolbar>  
    <e-items>  
      <e-item text='Cut'></e-item>  
      <e-item text='Copy'></e-item>  
      <e-item text='Paste'></e-item>  
      <e-item type='Separator'></e-item>  
      <e-item text='Bold'></e-item>  
      <e-item text='Italic'></e-item>  
      <e-item text='Underline'></e-item>  
    </e-items>  
  </ejs-toolbar>`  
})  
  
export class AppComponent {
```



```
}
`
```

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

```
`shell
npm start
`
```

The following code example depicts the way to initialize The Toolbar on a single element.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToolbarModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild } from '@angular/core';
import { ToolbarComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    ToolbarModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-toolbar>
      <e-items>
        <e-item text='Cut'></e-item>
        <e-item text='Copy'></e-item>
        <e-item text='Paste'></e-item>
        <e-item type='Separator'></e-item>
        <e-item text='Bold'></e-item>
        <e-item text='Italic'></e-item>
        <e-item text='Underline'></e-item>
      </e-items>
    </ejs-toolbar>
  `
})
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));
```

Initialize the Toolbar using HTML elements

The Toolbar component can be rendered based on the given HTML element using `<ejs-toolbar>`.

You need to follow the below structure of HTML elements to render the Toolbar inside the `<ejs-toolbar>` tag.

```
`html
<ejs-toolbar> --> Root Toolbar Element
<div> --> Toolbar Items Container
<div> --> Toolbar Item Element
</div>
</div>
</ejs-toolbar>
`
```

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToolbarModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild } from '@angular/core';
import { ToolbarComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    ToolbarModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-toolbar>
      <div>
        <div><button class='e-btn e-tbar-btn'>Cut</button> </div>
        <div><button class='e-btn e-tbar-btn'>Copy</button> </div>
        <div><button class='e-btn e-tbar-btn'>Paste</button> </div>
        <div class='e-separator'> </div>
        <div><button class='e-btn e-tbar-btn'>Bold</button> </div>
        <div><button class='e-btn e-tbar-btn'>Italic</button> </div>
      </div>
    </ejs-toolbar>
  `
})
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

- [How to add Toggle Button](#)

Note: You can refer to our [Angular Toolbar](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Toolbar Example](#) that shows you how to render the Toolbar in Angular.

Item configuration in Angular Toolbar component

The Toolbar can be rendered by defining an array of [items](#). Items can be constructed with the following built-in command types or item template.

Button

Button is the default command [type](#), and it can be rendered by using the [text](#) property.

Properties of the button command type:

Property | Description

[text](#) | The text to be displayed for button.

[id](#) | The ID of the button to be rendered. If the ID is not given, auto ID is generated.

[prefixIcon](#) | Defines the class used to specify an icon for the button. The icon is positioned before the text if text is available or the icon alone button is rendered.

[suffixIcon](#) | Defines the class used to specify an icon for the button. The icon is positioned after the text if text is available. If both [prefixIcon](#) and [suffixIcon](#) are specified, only [prefixIcon](#) is considered.

[width](#) | Used to set the [width](#) of the button.

[align](#) | Specifies the location for aligning Toolbar items.

Separator

The **Separator** type adds a vertical separation between the Toolbar's single/multiple commands.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToolbarModule } from '@syncfusion/ej2-angular-navigations'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { Component, ViewChild } from '@angular/core';
import { ToolbarComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    ToolbarModule, TooltipModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-toolbar>
      <e-items>
        <e-item text='Cut'></e-item>
        <e-item text='Copy'></e-item>
        <e-item type='Separator'></e-item>
        <e-item text='Paste'></e-item>
        <e-item type='Separator'></e-item>
        <e-item text='Undo'></e-item>
        <e-item text='Redo'></e-item>
      </e-items>
    </ejs-toolbar>`
})
```

```
  })  
  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));
```

If **Separator** is added as first or last item, it is not visible.

Input

The **Input** type is only applicable for adding **template** elements when the **template** property is defined as an **object**.

Input type creates an **input element** internally that acts as the container for **Syncfusion** input based components.

Note: Set toolbar item **type** property value as **Input** only for Input components.

NumericTextBox

- The **NumericTextBox** component can be included by importing the **NumericTextBox** module from ej2-inputs.
- Initialize the **NumericTextBox** in template property, in which the Toolbar item type set as **Input**.
- Related **NumericTextBox** component properties are also can be configured like as below.

`javascript

```
new NumericTextBox( { format: 'c2' })
```

,

DropDownList

- The **DropDownList** component can be included by importing the **DropDownList** module from ej2-dropdowns.
- Initialize the **DropDownList** in template property, in which the Toolbar item type set as **Input**.
- Related **DropDownList** component properties are also can be configured like as below.

`javascript

```
new DropDownList({ width:100 })
```

,

CheckBox

- The **CheckBox** component can be included by importing the **CheckBox** module from **ej2-buttons**.
- Initialize the **CheckBox** in template property, in which the Toolbar item type set as **Input**.
- Related **CheckBox** component properties are also can be configured like as below.

```
`javascript
```

```
new CheckBox({ label: 'Checkbox', checked: true })
```

```
,
```

RadioButton

- The **RadioButton** component can be included by importing the **RadioButton** module from **ej2-buttons**.
- Initialize the **RadioButton** in template property, in which the Toolbar item type set as **Input**.
- Related **RadioButton** component properties are also can be configured like as below.

```
`javascript
```

```
new RadioButton({ label: 'Radio', name: 'default', checked: true })
```

```
,
```

Above steps applicable for all 'Syncfusion' input based components.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToolbarModule } from '@syncfusion/ej2-angular-navigations'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { NumericTextBoxModule } from '@syncfusion/ej2-angular-inputs'
import { RadioButtonModule } from '@syncfusion/ej2-angular-buttons'
import { CheckBoxModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, ViewChild } from '@angular/core';
import { ToolbarComponent } from '@syncfusion/ej2-angular-navigations';
import { NumericTextBox } from '@syncfusion/ej2-inputs';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { CheckBox, RadioButton } from '@syncfusion/ej2-buttons';
@Component({
  imports: [
    ToolbarModule, TooltipModule, NumericTextBoxModule,
    RadioButtonModule, CheckBoxModule, DropDownListModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-toolbar>
      <e-items>
        <e-item text='Cut'></e-item>
        <e-item text='Copy'></e-item>
      </e-items>
    </ejs-toolbar>`
})
```

```

        <e-item type='Separator'></e-item>
        <e-item text='Undo'></e-item>
        <e-item text='Redo'></e-item>
        <e-item type='Input'>
            <ng-template #template>
                <ejs-numerictextbox format="c2" value="1"
width="150"></ejs-numerictextbox>
            </ng-template>
        </e-item>
        <e-item type='Input'>
            <ng-template #template>
                <ejs-dropdownlist [dataSource]='this.data' width="120"
index="2"></ejs-dropdownlist>
            </ng-template>
        </e-item>
        <e-item type='Input'>
            <ng-template #template>
                <ejs-checkbox label="CheckBox: true" [checked]="true"></ejs-
checkbox>
            </ng-template>
        </e-item>
        <e-item type='Input'>
            <ng-template #template>
                <ejs-radiobutton label='Radio' name='default'
checked="true"></ejs-radiobutton>
            </ng-template>
        </e-item>
    </e-items>
</ejs-toolbar>
    })
    export class AppComponent {
        @ViewChild('element') element?: any;
        public data: string[] = ['Badminton', 'Basketball', 'Cricket', 'Golf',
'Hockey', 'Rugby'];
    }

```

MAIN.TS

```

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

```

Enabling tab key navigation in Toolbar

The [tabIndex](#) property of a Toolbar item is used to enable tab key navigation for the item. By default, the user can switch between items using the arrow keys, but the [tabIndex](#) property allows you to switch between items using the Tab and Shift+Tab keys as well.

To use the [tabIndex](#) property, you need to set it for each Toolbar item that you want to enable tab key navigation. The [tabIndex](#) property should be set to a positive integer value. A value of 0 or a negative value will disable tab key navigation for the item.

For example, to enable tab key navigation for two Toolbar items, you can use the following code:

```
`javascript
import { Component, ViewChild } from '@angular/core';
import { ToolbarComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  selector: 'app-container',
  template: `
    <ejs-toolbar>
    <e-items>
    <e-item text='item1' tabIndex=1></e-item>
    <e-item text='item2' tabIndex=2></e-item>
    </e-items>
    </ejs-toolbar>
  `
})
export class AppComponent {
}
```

With the above code, the user can switch between the two Toolbar items using the Tab and Shift+Tab keys, in addition to using the arrow keys. The items will be navigated in the order specified by the [tabIndex](#) values.

If you set the [tabIndex](#) value to 0 for all Toolbar items, tab key navigation will be based on the element order rather than the [tabIndex](#) values. For example:

```
`javascript
import { Component, ViewChild } from '@angular/core';
import { ToolbarComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  selector: 'app-container',
  template: `
    <ejs-toolbar>
    <e-items>
    <e-item text='item1' tabIndex=0></e-item>
    <e-item text='item2' tabIndex=0></e-item>
    </e-items>
    </ejs-toolbar>
  `
})
export class AppComponent {
}
```

```

、
}))
export class AppComponent {
}
、

```

In this case, the user can switch between the two Toolbar items using the Tab and Shift+Tab keys, and the items will be navigated in the order in which they appear in the DOM.

Example:

Here is an example of how you can use the [tabIndex](#) property to enable tab key navigation for a Toolbar component:

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToolbarModule } from '@syncfusion/ej2-angular-navigations'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { Component, ViewChild } from '@angular/core';
import { ToolbarComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    ToolbarModule, TooltipModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-toolbar>
      <e-items>
        <e-item text='Cut' tabIndex=0></e-item>
        <e-item text='Copy' tabIndex=0></e-item>
        <e-item type='Separator'></e-item>
        <e-item text='Paste' tabIndex=0></e-item>
        <e-item type='Separator'></e-item>
        <e-item text='Undo' tabIndex=0></e-item>
        <e-item text='Redo' tabIndex=0></e-item>
      </e-items>
    </ejs-toolbar>
  `
})
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));

```


With the above code, the user can switch between the Toolbar items using the Tab and Shift+Tab keys, and the items will be navigated based on the element order.

See Also

- [How to set item wise custom template](#)

Template configuration in Angular Toolbar component

Integrate menu component

You can integrate menu component as toolbar item in Toolbar using **ng-template**. Menu can be populated with items as needed.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { ToolbarAllModule, MenuModule } from '@syncfusion/ej2-angular-navigations'
import { ButtonAllModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
import { ToolbarComponent, MenuItemModel } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    FormsModule,
    ToolbarAllModule,

    MenuModule,
    ButtonAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-toolbar>
    <e-items>
      <e-item text='Cut'></e-item>
      <e-item text='Copy'></e-item>
      <e-item text='Paste'></e-item>
      <e-item>
        <ng-template #template>
          <ejs-menu [items]="data"></ejs-menu>
        </ng-template>
      </e-item>
    </e-items>
  </ejs-toolbar>`
})
export class AppComponent {
  public data: MenuItemModel[] = [
    {
      text: 'Appliances',
      items: [
        {
          text: 'Kitchen',
          items: [
            { text: 'Electric Cookers' },

```

```

        { text: 'Coffee Makers' },
        { text: 'Blenders' },
    ],
},
{
    text: 'Washing Machine',
    items: [{ text: 'Fully Automatic' }, { text: 'Semi Automatic' }],
},
{
    text: 'Air Conditioners',
    items: [
        { text: 'Inverter ACs' },
        { text: 'Split ACs' },
        { text: 'Window ACs' },
    ],
},
],
},
{
    text: 'Accessories',
    items: [
        {
            text: 'Mobile',
            items: [
                { text: 'Headphones' },
                { text: 'Memory Cards' },
                { text: 'Power Banks' },
            ],
        },
    ],
},
{
    text: 'Computer',
    items: [
        { text: 'Pendrives' },
        { text: 'External Hard Disks' },
        { text: 'Monitors' },
    ],
},
],
},
{
    text: 'Fashion',
    items: [
        {
            text: 'Men',
            items: [
                { text: 'Shirts' },
                { text: 'Jackets' },
                { text: 'Track Suits' },
            ],
        },
        {
            text: 'Women',
            items: [{ text: 'Kurtas' }, { text: 'Salwars' }, { text: 'Sarees' }],
        },
    ],
},
],
},

```

```
{
  text: 'Home & Living',
  items: [
    {
      text: 'Furniture',
      items: [
        { text: 'Beds' },
        { text: 'Mattresses' },
        { text: 'Dining Tables' },
      ],
    },
    {
      text: 'Decor',
      items: [
        { text: 'Clocks' },
        { text: 'Wall Decals' },
        { text: 'Paintings' },
      ],
    },
  ],
},
];
}
```

MAIN.TS

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

Responsive mode in Angular Toolbar component

This section explains the supported display modes of the Toolbar when the content exceeds the viewing area. Possible modes are:

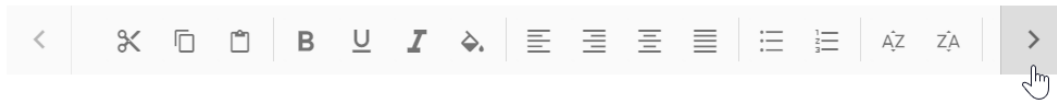
- Scrollable
- Popup

Scrollable

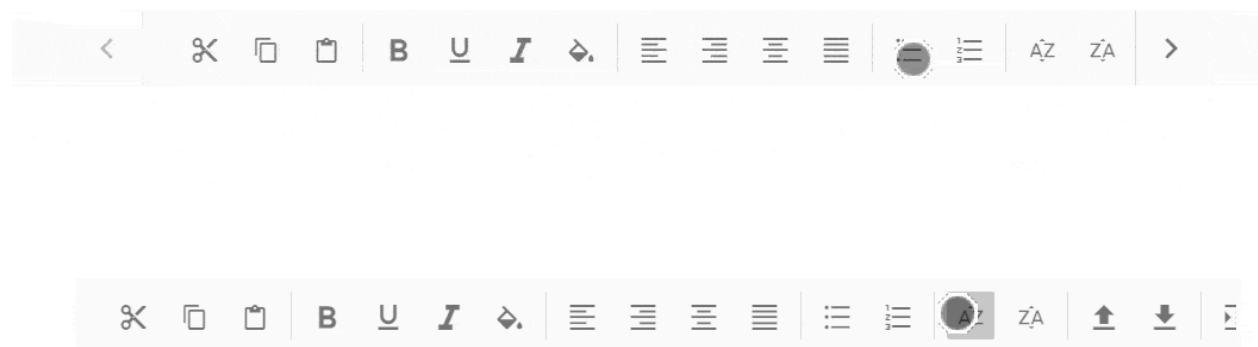
The default overflow mode of the Toolbar is **Scrollable**. Scrollable display mode supports display of commands in a single line with horizontal scrolling enabled when commands overflow to available space.

- The right and left navigation arrows are added to the start and end of the Toolbar to navigate to hidden commands.
- You can also see the hidden commands using touch swipe action.
- By default, if navigation icon in the **left** side is disabled, you can see the hidden commands by moving to the **right**.
- By clicking the arrow or by holding the arrow continuously, hidden commands will become visible.

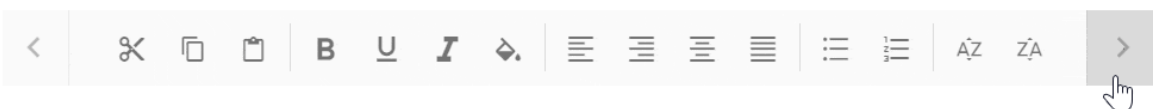
- If device navigation icons are not available, you can touch swipe to see the hidden commands of the Toolbar.



- Once the Toolbar reaches the last or first command, the corresponding navigation icon will be disabled and you can move to the opposite direction.



- You can continuously scroll the Toolbar content by holding the navigation icon.



APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToolbarModule } from '@syncfusion/ej2-angular-navigations'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { Component, ViewChild } from '@angular/core';
import { ToolbarComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    ToolbarModule, TooltipModule
```

```

    ],
    standalone: true,
    selector: 'app-container',
    template: `
      <ejs-toolbar width="300px">
        <e-items>
          <e-item text='Cut' prefixIcon = 'e-cut-icon tb-icons'></e-item>
          <e-item text='Copy' prefixIcon = 'e-copy-icon tb-icons'></e-
item>
          <e-item text='Paste' prefixIcon = 'e-paste-icon tb-icons'></e-
item>
          <e-item type='Separator'></e-item>
          <e-item text='bold' prefixIcon = 'e-bold-icon tb-icons'></e-
item>
          <e-item text='underline' prefixIcon = 'e-underline-icon tb-
icons'></e-item>
          <e-item text='italic' prefixIcon = 'e-italic-icon tb-
icons'></e-item>
          <e-item type='Separator'></e-item>
          <e-item text='A-Z Sort' prefixIcon = 'e-ascending-icon tb-
icons'></e-item>
          <e-item text='Z-A Sort' prefixIcon = 'e-descending-icon tb-
icons'></e-item>
          <e-item text='Clear' prefixIcon = 'e-clear-icon tb-icons'></e-
item>
        </e-items>
      </ejs-toolbar>
    `
  })
  export class AppComponent {
  }

```

MAIN.TS

```

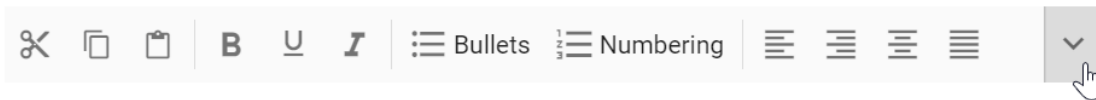
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Popup

Popup is another type of [overflowMode](#) in which the Toolbar container holds the commands that can be placed in the available space. The rest of the overflowing commands that do not fit within the viewing area moves to the overflow popup container.

The commands placed in the popup can be viewed by opening the popup using the drop down icon given at the end of the Toolbar.



If the popup content overflows the height of the page, then the rest of the commands will be hidden.

Priority of commands

Default popup priority is set as `none`, and when the commands of the Toolbar overflow, the ones listed last will be moved to the popup.

You can customize the priority of commands to be displayed on the Toolbar and popup by using the [overflow](#) property.

Property | Description

Both | Button Text is visible in both **Toolbar** and **Popup**.

Overflow | Button Text is only visible in **Popup**.

Toolbar | Button Text is only visible on the **Toolbar**.

In the following code sample, text is only visible in the popup container and not in the Toolbar container.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToolbarModule } from '@syncfusion/ej2-angular-navigations'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { Component, ViewChild } from '@angular/core';
import { ToolbarComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    ToolbarModule, TooltipModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-toolbar overflowMode = 'Popup' width= 330>
      <e-items>
        <e-item text='Cut' prefixIcon = 'e-cut-icon tb-icons' overflow
= 'Show' showTextOn = 'overflow'></e-item>
        <e-item text='Copy' prefixIcon = 'e-copy-icon tb-icons'
overflow = 'Show' showTextOn = 'overflow'></e-item>
        <e-item text='Paste' prefixIcon = 'e-paste-icon tb-icons'
overflow = 'Show' showTextOn = 'overflow'></e-item>
      </e-items>
    </ejs-toolbar>
  `
})
export class AppComponent {
  title = 'app';
}
```

```

        <e-item type='Separator'></e-item>
        <e-item text='bold' prefixIcon = 'e-bold-icon tb-icons'
overflow = 'Show' showTextOn = 'overflow' ></e-item>
        <e-item text='underline' prefixIcon = 'e-underline-icon tb-
icons' overflow = 'Show' showTextOn = 'overflow' ></e-item>
        <e-item text='italic' prefixIcon = 'e-italic-icon tb-icons'
overflow = 'Show' showTextOn = 'overflow' ></e-item>
        <e-item text='Color-Picker' prefixIcon = 'e-color-icon tb-
icons' overflow = 'Hide' showTextOn = 'overflow' ></e-item>
        <e-item type='Separator'></e-item>
        <e-item text='A-Z Sort' prefixIcon = 'e-ascending-icon tb-
icons' overflow = 'Show' showTextOn = 'overflow'></e-item>
        <e-item text='Z-A Sort' prefixIcon = 'e-descending-icon tb-
icons' overflow = 'Show' showTextOn = 'overflow'></e-item>
        <e-item text='Clear' prefixIcon = 'e-clear-icon tb-icons'
overflow = 'Show' showTextOn = 'overflow'></e-item>
    </e-items>
</ejs-toolbar>
`
    })
    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 toolbar with a less than minimum height

By default, the toolbar renders with a minimum height to render items properly. You can also render a customized toolbar with a height that is less than the minimum height of the toolbar by overriding default styles from the application end.

APP.COMPONENT.HTML

```

<div class="control-section e-tbar-section">
  <div class="e-sample-resize-container">
    <h3>Scrollable mode </h3>
    <!-- Render the Toolbar Component -->
    <ejs-toolbar height="20px" width="300px">
      <e-items>
        <e-item text='Cut' prefixIcon='e-icons e-cut'></e-item>
        <e-item text='Copy' prefixIcon='e-icons e-copy'></e-item>
        <e-item text='Paste' prefixIcon='e-icons e-paste'></e-item>
        <e-item type='Separator'></e-item>
        <e-item text='bold' prefixIcon='e-icons e-bold'></e-item>
        <e-item text='underline' prefixIcon='e-icons e-underline'></e-item>
        <e-item text='italic' prefixIcon='e-icons e-italic'></e-item>
        <e-item type='Separator'></e-item>
        <e-item text='Export' prefixIcon='e-icons e-export'></e-item>
        <e-item text='Reload' prefixIcon='e-icons e-refresh'></e-item>
        <e-item text='Clear' prefixIcon='e-icons e-erase'></e-item>
      </e-items>
    </ejs-toolbar>
  </div>
</div>

```

```

    </e-items>
  </ejs-toolbar>
  <br>
  <h3>Popup mode </h3>
  <!-- Render the Toolbar Component -->
  <ejs-toolbar height="20px" overflowMode='Popup' width='380px'>
    <e-items>
      <e-item text='Cut' prefixIcon='e-icons e-cut' overflow='Show'
showTextOn='overflow'></e-item>
      <e-item text='Copy' prefixIcon='e-icons e-copy' overflow='Show'
showTextOn='overflow'></e-item>
      <e-item text='Paste' prefixIcon='e-icons e-paste' overflow='Show'
showTextOn='overflow'></e-item>
      <e-item type='Separator'></e-item>
      <e-item text='bold' prefixIcon='e-icons e-bold' overflow='Show'
showTextOn='overflow'></e-item>
      <e-item text='underline' prefixIcon='e-icons e-underline'
overflow='Show' showTextOn='overflow'></e-item>
      <e-item text='italic' prefixIcon='e-icons e-italic' overflow='Show'
showTextOn='overflow'></e-item>
      <e-item text='Color-Picker' prefixIcon='e-icons e-paint-bucket'
overflow='Hide' showTextOn='overflow'></e-item>
      <e-item type='Separator'></e-item>
      <e-item text='Export' prefixIcon='e-icons e-export' overflow='Show'
showTextOn='overflow'></e-item>
      <e-item text='Reload' prefixIcon='e-icons e-refresh' overflow='Show'
showTextOn='overflow'></e-item>
      <e-item text='Clear' prefixIcon='e-icons e-erase' overflow='Show'
showTextOn='overflow'></e-item>
    </e-items>
  </ejs-toolbar>
</div>
</div>

```

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToolbarModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewEncapsulation } from '@angular/core';
import { ToolbarComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    ToolbarModule
  ],
  standalone: true,
  selector: 'app-container',
  templateUrl: './app.component.html',
  encapsulation: ViewEncapsulation.None
})
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));
```

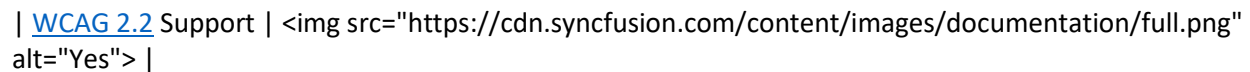
Accessibility in Angular Toolbar component

The [Angular Toolbar](#) 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-label`, and `aria-orientation`. 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 Toolbar component is outlined below.

| Accessibility Criteria | Compatibility |

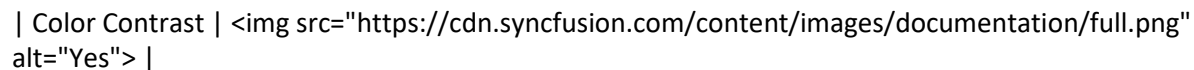
| -- | -- |

| [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>

ARIA attributes

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

| Property | Functionalities |

| --- | --- |

| role="toolbar" | Attribute is set to the ToolBar element describes the actual role of the element. |

| aria-orientation | Attribute is set to the ToolBar element to indicates the ToolBar orientation. Default value is **horizontal**. |

| aria-label | Attribute is set to ToolBar element describes the purpose of the set of toolbar. |

| aria-expanded | Attribute is set to the ToolBar Popup element to indicates the expanded state of the popup. |

| aria-haspopup | Attribute is set to the popup element to indicates the popup mode of the Toolbar. Default value is false. When popup mode is enabled, attribute value has to be changed to **true**. |

| aria-disabled | Attribute set to the ToolBar element to indicates the disabled state of the ToolBar. |

Keyboard interaction

Keyboard navigation is enabled by default. Possible keys are

Key	Description
-----	-----

| Left | Focuses the previous element. |

| Right | Focuses the next element. |

| Enter | When focused on a ToolBar command, clicking the key triggers the click of Toolbar element. When popup drop-down icon is focused, the popup opens. |

| Esc(Escape) | Closes popup. |

| Down | Focuses the next popup element. |

| Up | Focuses the previous popup element. |

| Home | Moves focus to the first Toolbar. |

| End | Moves focus to the last Toolbar. |

| Tab | To Move focus through the interactive elements. |

| Shift + Tab | To Move focus through the interactive elements. |

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion Angular components](#)

Style in Angular Toolbar component

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

Customizing Toolbar

Use the following CSS to customize the Toolbar.

```
`CSS
.e-toolbar {
border: 5px solid rgb(173, 255, 47);
}
`
```

Customizing the Toolbar items

Use the following CSS to customize the items of Toolbar.

```
`CSS
.e-toolbar .e-toolbar-item {
background: #add8e6;
border: 1px solid #5a70cc;
}
`
```

Use the following CSS to customize the button in the items of Toolbar.

```
`CSS
.e-toolbar .e-tbar-btn {
background: #add8e6;
border: 1px solid #5a70cc;
}
`
```

Customizing Toolbar's item icon

Use the following CSS to customize the item icon of Toolbar control.

```
`CSS
.e-toolbar .e-tbar-btn .e-icons {
background: #185655;
color: #d7f9d4;
}
`
```

Customizing the hover state of Toolbar control

Use the following CSS to customize the toolbar item when hovering.

```
`CSS
.e-toolbar .e-tbar-btn:hover {
background: #c0e3a1;
border: 1px solid green;
}
`
```

Customizing selected item of Toolbar control

Use the following CSS to customize the selected toolbar item.

```
`CSS
.e-toolbar .e-tbar-btn:focus {
background: #add8e6;
border: 1px solid #5a70cc;
}
`
```

How To

Set command customization in Angular Toolbar component

The [htmlAttributes](#) property of the Toolbar item is used to set the HTML attributes ('ID', 'class', 'style', 'role') for the commands.

When style attributes are added, if the same attributes were already present, they will be replaced. This is not so in the case of `class` attribute. Classes will be added to the element instead of replacing the existing ones.

Single or multiple CSS classes can be added to the Toolbar commands using the Toolbar item [cssClass](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToolbarModule } from '@syncfusion/ej2-angular-navigations'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { Component, ViewChild } from '@angular/core';
```

```
import { ToolbarComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    ToolbarModule, TooltipModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-toolbar>
      <e-items>
        <e-item text='Bold' [htmlAttributes] = 'boldAttribute'></e-
item>
        <e-item text='Italic' [htmlAttributes] = 'italicAttribute'></e-
item>
        <e-item text='Underline' [htmlAttributes] =
'underAttribute'></e-item>
        <e-item type='Separator'></e-item>
        <e-item text='Uppercase' cssClass = 'e-txt-casing'></e-item>
      </e-items>
    </ejs-toolbar>
  `
})
export class AppComponent {
  @ViewChild('element') element?: any;
  public boldAttribute: any = { 'class': 'custom_bold', 'id': 'itemId' };
  public italicAttribute: any = { 'class': 'custom_italic' };
  public underAttribute: any = { 'class': 'custom_underline' };
  ngAfterViewInit() {
  }
}
```

MAIN.TS

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

Set tool tip to the commands in Angular Toolbar component

The [tooltipText](#) property of the Toolbar item is used to set the HTML Tooltip to the commands that can be viewed as hint texts on mouse hover.

To change the [tooltipText](#) to ej2-tooltip component:

- Import the **Tooltip** module from **ej2-popups**, and initialize the Tooltip with the Toolbar target. Refer to the following code example:

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ToolbarModule } from '@syncfusion/ej2-angular-navigations';
import { TooltipModule } from '@syncfusion/ej2-angular-popups';
import { Component, ViewChild } from '@angular/core';
```

```

@Component({
  imports: [
    ToolbarModule, TooltipModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-tooltip id="Tooltip" target="#Toolbar [title]">
      <ejs-toolbar id='Toolbar'>
        <e-items>
          <e-item text='Cut' tooltipText = 'Cut'></e-item>
          <e-item text='Copy' tooltipText = 'Copy'></e-item>
          <e-item text='Paste' tooltipText = 'Paste'></e-item>
          <e-item text='Undo' tooltipText = 'Undo'></e-item>
          <e-item text='Redo' tooltipText = 'Redo'></e-item>
        </e-items>
      </ejs-toolbar>
    </ejs-tooltip>
  `
})
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));

```

Set item wise custom template in Angular Toolbar component

The Toolbar supports adding template commands using the `template` property. Template property can be given as the `HTML element` that is either a `string` or a `query selector`.

As string

The HTML element tag can be given as a string for the template property. Here, the checkbox is rendered as a HTML template.

`typescript

```
template: "<div><input type='checkbox' id='check1' checked=''>Accept</input></div>"
```

,

As selector

The template property also allows getting template content through query `selector`. Here, button 'ID' attribute is specified in the template.

`typescript

```
template: "#Template"
```

,

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToolbarModule } from '@syncfusion/ej2-angular-navigations'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    ToolbarModule, TooltipModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `
    <button id='Template' class='e-btn'>Template</button>
    <ejs-toolbar>
      <e-items>
        <e-item text='Cut'></e-item>
        <e-item type='Separator'></e-item>
        <e-item text='Paste' prefixIcon = 'e-paste-icon'></e-item>
        <e-item type='Separator'></e-item>
        <e-item [template]='templateEle'></e-item>
        <e-item text='Undo'></e-item>
        <e-item text='Redo'></e-item>
        <e-item [template]='templateEleId'></e-item>
      </e-items>
    </ejs-toolbar>
  `
})
export class AppComponent {
  @ViewChild('element') element?: any;
  public templateEle: any = "<div><input type='checkbox' id='check1' checked='\"'>Accept</input></div>";
  public templateEleId: any = '#Template';
  ngAfterViewInit() {
  }
}

```

MAIN.TS

```

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

```

Add toggle button in Angular Toolbar component

Toolbar supports to add a toggle Button by using the [template](#) property. Refer below steps

- By using Toolbar template property, pass required HTML String to render toggle button.

`typescript

```
<e-item template='<button class="e-btn" id="media_btn"></button>'></e-item>
```

,

- Now render the toggle Button into the targeted element in Toolbar [created](#) event handler and bind click event for it. On clicking the toggle Button, change the required icon and content based on current active state.

APP.COMPONENT.HTML

```

<div class="control-section e-tbar-section">
  <div class="e-sample-resize-container">
    <!-- Render the Toolbar Component -->
    <ejs-toolbar id="toolbar" class="toggle">
      <e-items>
        <e-item>
          <ng-template #template>
            <button ejs-button class="e-btn" cssClass="e-flat"
iconCss="e-icons e-play-icon" isToggle="true" #playButton
(click)="mediaBtnClick()"></button>
          </ng-template>
        </e-item>
        <e-item type='Separator'></e-item>
        <e-item>
          <ng-template #template>
            <button ejs-button class="e-btn" cssClass="e-flat"
iconCss="e-icons e-zoomin-icon" isToggle="true" #zoom
(click)="zoomBtnClick()"></button>
          </ng-template>
        </e-item>
        <e-item type='Separator'></e-item>
        <e-item>
          <ng-template #template>
            <button ejs-button class="e-btn" cssClass="e-flat"
iconCss="e-icons e-undo-icon" isToggle="true" #undo
(click)="undoBtnClick()"></button>
          </ng-template>
        </e-item>
        <e-item type='Separator'></e-item>
        <e-item>
          <ng-template #template>
            <button ejs-button class="e-btn" cssClass="e-flat"
iconCss="e-icons e-filter-icon" isToggle="true" #filter
(click)="filterBtnClick()"></button>
          </ng-template>
        </e-item>
        <e-item type='Separator'></e-item>
        <e-item>
          <ng-template #template>
            <button ejs-button class="e-btn" cssClass="e-flat"
iconCss="e-icons e-hide-icon" isToggle="true" #hide (click)="hideBtnClick()"
[content]="content"></button>
          </ng-template>
        </e-item>
      </e-items>
    </ejs-toolbar>
    <br/>
    <div id="content">
      This content will be hidden, when you click on hide button and
toggle get an active state as show, otherwise it will be visible.
    </div>
  </div>

```



```

        </div>
    </div>
</div>

```

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToolbarModule } from '@syncfusion/ej2-angular-navigations'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewEncapsulation, Inject, ViewChild } from
'@angular/core';
import { Button, ButtonComponent } from '@syncfusion/ej2-angular-buttons';
import { ToolbarComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    ToolbarModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-container',
  templateUrl: './app.component.html',
  encapsulation: ViewEncapsulation.None
})
export class DefaultToolbarComponent {
  @ViewChild('playButton') public mediaBtn!: ButtonComponent;
  @ViewChild('zoom') public zoomBtn!: ButtonComponent;
  @ViewChild('undo') public undoBtn!: ButtonComponent;
  @ViewChild('filter') public filterBtn!: ButtonComponent;
  @ViewChild('hide') public visibleBtn!: ButtonComponent;

  public content: string = 'Hide';
  public mediaBtnClick() {
    if (this.mediaBtn.element.classList.contains('e-active')) {
      this.mediaBtn.iconCss = 'e-icons e-play-icon';
    } else {
      this.mediaBtn.iconCss = 'e-icons e-pause-icon';
    }
  }
  public zoomBtnClick() {
    if (this.zoomBtn.element.classList.contains('e-active')) {
      this.zoomBtn.iconCss = 'e-icons e-zoomin-icon';
    } else {
      this.zoomBtn.iconCss = 'e-icons e-zoomout-icon';
    }
  }
  public undoBtnClick() {
    if (this.undoBtn.element.classList.contains('e-active')) {
      this.undoBtn.iconCss = 'e-icons e-undo-icon';
    } else {
      this.undoBtn.iconCss = 'e-icons e-redo-icon';
    }
  }
  public filterBtnClick() {
    if (this.filterBtn.element.classList.contains('e-active')) {
      this.filterBtn.iconCss = 'e-icons e-filter-icon';
    } else {

```

```

        this.filterBtn.iconCss = 'e-icons e-filternone-icon';
    }
}
public hideBtnClick() {
    if (this.visibleBtn.element.classList.contains('e-active')) {
        (document.getElementById('content') as
HTMLElement).style.display = 'block';
        this.visibleBtn.iconCss = 'e-icons e-hide-icon';
        this.content = 'Hide';
    } else {
        (document.getElementById('content') as
HTMLElement).style.display = 'none';
        this.visibleBtn.iconCss = 'e-icons e-show-icon';
        this.content = 'Show';
    }
}
}
}

```

MAIN.TS

```

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

```

Add font awesome in Angular Toolbar component

You can customize the Toolbar component items by using third-party icons other than the icons available in the Syncfusion library. In the following example, font awesome icons are used as toolbar items.

- Refer to the third-party reference link. Here, the CDN link of font awesome is referred.

```
`html
```

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css" />
```

```
,
```

- Add the icons to the toolbar component using '[prefixIcon](#)' property

The following sample explains how to use font awesome in the toolbar component.

APP.COMPONENT.TS

```

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

```

```

standalone: true,
  selector: 'app-container',
  template: `
    <ejs-toolbar>
      <e-items>
        <e-item prefixIcon="fa fa-twitter"></e-item>
        <e-item prefixIcon="fa fa-facebook"></e-item>
        <e-item prefixIcon="fa fa-whatsapp"></e-item>
        <e-item
          template='<button class="e-btn e-tbar-btn"><span><i style="font-size: 3em !important; color: Tomato" class="e-icons fa fa-twitter"></i></span></button>'></e-item>
        </e-items>
      </ejs-toolbar>`
  })
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));

```

We can also use templates for rendering icons based on the requirements.

Customization

The class “e-icons” is used for standardizing the appearance of the icons to fit into toolbar items. If you wish to override the appearance of the icons used, you could do this by using the following set of classes

Use the following CSS to set the color of icons.

`CSS

```

.e-toolbar .e-icons {
  color: #e3165b !important;
}
`

```

Use the following CSS to set the font size of icons.

`CSS

```

.e-toolbar .e-btn .e-icons.e-btn-icon {
  font-size: 14px !important;
}
`

```

How to customize toolbar scroll step in Angular Toolbar component

Toolbar supports to customize the scrolling distance when you click the left and right side navigation icons. we can customize `ScrollStep` property for scrolling distance. Refer to the following code example.

By using Toolbar scrollStep property, pass a required value to customize toolbar scrollStep.

APP.COMPONENT.HTML

```
<div class="control-section e-tbar-section">
  <div class="e-sample-resize-container">
    <!-- Render the Toolbar Component -->
    <ejs-toolbar id="toolbar" scrollStep="50">
      <e-items>
        <e-item prefixIcon='e-cut-icon tb-icons'
tooltipText='Cut'></e-item>
        <e-item prefixIcon='e-copy-icon tb-icons'
tooltipText='Copy'></e-item>
        <e-item prefixIcon='e-paste-icon tb-icons'
tooltipText='Paste'></e-item>
        <e-item type='Separator'></e-item>
        <e-item prefixIcon='e-bold-icon tb-icons'
tooltipText='Bold'></e-item>
        <e-item prefixIcon='e-underline-icon tb-icons'
tooltipText='Underline'></e-item>
        <e-item prefixIcon='e-italic-icon tb-icons'
tooltipText='Italic'></e-item>
        <e-item prefixIcon='e-color-icon tb-icons'
tooltipText='Color-Picker'></e-item>
        <e-item type='Separator'></e-item>
        <e-item prefixIcon='e-alignleft-icon tb-icons'
tooltipText='Align-Left'></e-item>
        <e-item prefixIcon='e-alignright-icon tb-icons'
tooltipText='Align-Right'></e-item>
        <e-item prefixIcon='e-aligncenter-icon tb-icons'
tooltipText='Align-Center'></e-item>
        <e-item prefixIcon='e-alignjustify-icon tb-icons'
tooltipText='Align-Justify'></e-item>
        <e-item type='Separator'></e-item>
        <e-item prefixIcon='e-bullets-icon tb-icons'
tooltipText='Bullets'></e-item>
        <e-item prefixIcon='e-numbering-icon tb-icons'
tooltipText='Numbering'></e-item>
        <e-item type='Separator'></e-item>
        <e-item prefixIcon='e-ascending-icon tb-icons'
tooltipText='Ascending'></e-item>
        <e-item prefixIcon='e-descending-icon tb-icons'
tooltipText='Descending'></e-item>
        <e-item type='Separator'></e-item>
        <e-item prefixIcon='e-upload-icon tb-icons'
tooltipText='Upload'></e-item>
        <e-item prefixIcon='e-download-icon tb-icons'
tooltipText='Download'></e-item>
        <e-item type='Separator'></e-item>
        <e-item prefixIcon='e-indent-icon tb-icons'
tooltipText='Indent'></e-item>
        <e-item prefixIcon='e-outdent-icon tb-icons'
tooltipText='Outdent'></e-item>
        <e-item type='Separator'></e-item>
        <e-item prefixIcon='e-clear-icon tb-icons'
tooltipText='Clear'></e-item>
      </e-items>
    </ejs-toolbar>
  </div>
</div>
```

```

        <e-item prefixIcon='e-reload-icon tb-icons'
tooltipText='Reload'></e-item>
        <e-item prefixIcon='e-export-icon tb-icons'
tooltipText='Export'></e-item>
    </e-items>
</ejs-toolbar>
<br/>
</div>
</div>

```

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToolbarModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewEncapsulation, Inject } from '@angular/core';
import { ToolbarComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    ToolbarModule
  ],
  standalone: true,
  selector: 'app-container',
  templateUrl: './app.component.html',
  encapsulation: ViewEncapsulation.None
})
export class DefaultToolbarComponent {
}

```

MAIN.TS

```

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

```

Add link to toolbar item in Angular Toolbar component

You can add link inside Toolbar using Angular **ng-template**. We need to use **ng-template** inside the **e-item** tag with **#template** attribute, which is mandatory to render the template.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToolbarModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
import { ToolbarComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    ToolbarModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-toolbar>

```

```

        <e-items>
            <e-item text='Cut'></e-item>
            <e-item text='Copy'></e-item>
            <e-item type='Separator'></e-item>
            <e-item text='Paste'></e-item>
            <e-item type='Separator'></e-item>
            <e-item >
                <ng-template #template>
                    <div>
                        <a
                            target="_blank"
href="https://ej2.syncfusion.com/angular/documentation/toolbar/getting-
started/"
                            >Anchor Toolbar Link</a>
                        >
                    </div>
                </ng-template>
            </e-item>
        </e-items>
    </ejs-toolbar>
}
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 other components in toolbar using angular template in Angular Toolbar component

You can render other components inside Toolbar using Angular **ng-template**. Through this, we can add items as other components directly with all their functionalities to our Toolbar. We need to use **ng-template** inside the each **e-item** tag with **#template** attribute, which is mandatory to render that template.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { ToolbarAllModule } from '@syncfusion/ej2-angular-navigations'
import { DatePickerAllModule } from '@syncfusion/ej2-angular-calendars'
import { ButtonAllModule } from '@syncfusion/ej2-angular-buttons'
import { NumericTextBoxAllModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule,
    ToolbarAllModule,

    DatePickerAllModule,

```

```

        NumericTextBoxAllModule,
        ButtonAllModule
    ],
    standalone: true,
    selector: 'app-container',
    template: ` <ejs-toolbar>
        <e-items>
            <e-item >
                <ng-template #template>
                    <ejs-numerictextbox value="10"></ejs-numerictextbox>
                </ng-template>
            </e-item>
            <e-item >
                <ng-template #template>
                    <ejs-datepicker></ejs-datepicker>
                </ng-template>
            </e-item>
            <e-item text='Cut'></e-item>
            <e-item text='Copy'></e-item>
            <e-item text='Paste'></e-item>
            <e-item type='Separator'></e-item>
            <e-item text='Bold'></e-item>
            <e-item text='Italic'></e-item>
            <e-item text='Underline'></e-item>
        </e-items>
    </ejs-toolbar>`
    })
    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 Toolbar component

This article describes the API migration process of Toolbar component from Essential JS 1 to Essential JS 2.

Accessibility and Localization

<!-- markdownlint-disable MD033 -->

Behavior	Property in Essential JS 1	Property in Essential JS 2
Localization	Not Applicable	Property : locale <ejs-toolbar id='toolbar' locale='fr-BE'></ejs-toolbar>
Right to left	Property : enableRTL <ej-toolbar id="toolbar" [enableRTL]="true"></ej-toolbar>	Property : enableRTL <ejs-toolbar id='toolbar' [enableRTL]='true'> </ejs-toolbar>

DataSource

<!-- markdownlint-disable MD033 -->

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| DataSource | **Property** : dataSource
 <ej-toolbar id="toolbar" [dataSource]="items" ></ej-toolbar> | Not Applicable || Query | **Property** : query
 <ej-toolbar id="toolbar" [query]="query"></ej-toolbar> | Not Applicable || Fields | **Property** : fields
 <ej-toolbar id="toolbar" [fields]="fields"></ej-toolbar> | Not Applicable || Group | **Property** : group
 <ej-toolbar id="toolbar" [fields]="fields"></ej-toolbar>
 TS:
 public fields: Object = { group: " " } | Not Applicable || HtmlAttributes | **Property** : htmlAttributes
 <ej-toolbar id="toolbar" [fields]="fields"></ej-toolbar>
 TS:
 public fields: Object = { htmlAttributes: { } } | Not Applicable || Id | **Property** : id
 <ej-toolbar id="toolbar" [fields]="fields"></ej-toolbar>
 TS:
 public fields: Object = { id: " " } | Not Applicable || ImageAttributes | **Property** : imageAttributes
 <ej-toolbar id="toolbar" [fields]="fields" ></ej-toolbar>
 TS:
 public fields: Object = { imageAttributes: { } } | Not Applicable || ImageUrl | **Property** : imageUrl
 <ej-toolbar id="toolbar" [fields]="fields" ></ej-toolbar>
 TS:
 public fields: Object = { imageUrl: " " } | Not Applicable || SpriteCssClass | **Property** : spriteCssClass
 <ej-toolbar id="toolbar" [fields]="fields" ></ej-toolbar>
 TS:
 public fields: Object = { spriteCssClass: " " } | Not Applicable || Text | **Property** : text
 <ej-toolbar id="toolbar" [fields]="fields" ></ej-toolbar>
 TS:
 public fields: Object = { text: " " } | Not Applicable || TooltipText | **Property** : tooltipText
 <ej-toolbar id="toolbar" [fields]="fields" ></ej-toolbar>
 TS:
 public fields: Object = { tooltipText: " " } | Not Applicable || Template | **Property** : template
 <ej-toolbar id="toolbar" [fields]="fields" ></ej-toolbar>
 TS:
 public fields: Object = { template: " " } | Not Applicable |

Items

<!-- markdownlint-disable MD033 -->

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| Default | <**Property** : items
 <ej-toolbar id="toolbar" [items]="items"> </ej-toolbar> | **Property** : items
 <ejs-toolbar id="toolbar" [items]="items"> </ejs-toolbar> || Align | Not Applicable | **Property** : items[0].Align
 <ejs-toolbar id="toolbar" [items]="items"> </ejs-toolbar>
 TS:
 public items: Object[] = [{ align: 'center'}]; |

| Custom class | **Not Applicable** | **Property** : items[0].cssClass **
** **<ejs-toolbar id="toolbar" [items]="items"> </ejs-toolbar>
 TS:
** public items: Object[] = [{ cssClass: 'e-class'}]; |

| Group Name | **Property** : items[0].group **
** **<ej-toolbar id="toolbar" [items]="items"> </ej-toolbar>
 TS:
** public items: Object[] = [{ group: ''}]; | **Not Applicable** |

| HtmlAttributes | **Property** : items[0].htmlAttributes **
** **<ej-toolbar id="toolbar" [items]="items"> </ej-toolbar> </ej-toolbar>
 TS:
** public items: Object[] = [{ htmlAttributes: { } }]; | **Property** : items[0].htmlAttributes **
** **<ejs-toolbar id="toolbar" [items]="items"> </ej-toolbar> </ejs-toolbar>
 TS:
** public items: Object[] = [{ htmlAttributes: { } }]; |

| Id | **Property** : items[0].id **
** **<ej-toolbar id="toolbar" [items]="items"> </ej-toolbar>
 TS:
** public items: Object[] = [{ id: '' }]; | **Property** : items[0].id **
** **<ejs-toolbar id="toolbar" [items]="items"> </ejs-toolbar>
 TS:
** public items: Object[] = [{ id: '' }]; |

| ImageAttributes | **Property** : items[0].imageAttributes **
** **<ej-toolbar id="toolbar" [items]="items"> </ej-toolbar>
 TS:
** public items: Object[] = [{ imageAttributes: { } }]; | **Not Applicable** |

| ImageUrl | **Property** : items[0].imageUrl **
** **<ej-toolbar id="toolbar" [items]="items"> </ej-toolbar>
 TS:
** public items: Object[] = [{ imageUrl: '' }]; | **Not Applicable** |

| Overflow | **Not Applicable** | **Property** : items[0].overflow **
** **<ejs-toolbar id="toolbar" [items]="items"> </ejs-toolbar>
 TS:
** public items: Object[] = [{ overflow: 'popup' }]; |

| PrefixIcon | **Not Applicable** | **Property** : items[0].prefixIcon **
** **<ejs-toolbar id="toolbar" [items]="items"> </ejs-toolbar>
 TS:
** public items: Object[] = [{ prefixIcon: 'e-icon' }]; |

| ShowAlwaysInPopup | **Not Applicable** | **Property** : items[0].showAlwaysInPopup **
** **<ejs-toolbar id="toolbar" [items]="items"> </ejs-toolbar>
 TS:
** public items: Object[] = [{ showAlwaysInPopup: true }]; |

| ShowTextOn | **Not Applicable** | **Property** : items[0].showTextOn **
** **<ejs-toolbar id="toolbar" [items]="items"> </ejs-toolbar>
 TS:
** public items: Object[] = [{ showTextOn: 'popup' }]; |

| Sprite CssClass | **Property** : items[0].showTextOn **
** **<ej-toolbar id="toolbar" [items]="items"> </ej-toolbar>
 TS:
** public items: Object[] = [{ spriteCssClass: 'e-class' }]; | **Not Applicable** |

| SuffixIcon | **Not Applicable** | **Property** : items[0].suffixIcon **
** **<ejs-toolbar id="toolbar" [items]="items"> </ejs-toolbar>
 TS:
** public items: Object[] = [{ suffixIcon: 'e-icon' }]; |

| Template | **Property** : items[0].template **
** **<ej-toolbar id="toolbar" [items]="items"> </ej-toolbar>
 TS:
** public items: Object[] = [{ template: '' }]; | **Property** : items[0].template **
** **<ejs-toolbar id="toolbar" [items]="items"> </ejs-toolbar>
 TS:
** public items: Object[] = [{ template: '' }]; |

| Text | **Property** : items[0].text **
** **<ej-toolbar id="toolbar" [items]="items"> </ej-toolbar>
 TS:
** public items: Object[] = [{ text: 'Cut' }]; | **Property** : items[0].text **
** **<ejs-**

toolbar id="toolbar" [items]="items"> </ejs-toolbar>
 TS:
 public items: Object[] = [{ text: 'Copy' }]; |

| TooltipText | **Property** : items[0].tooltipText
 <ej-toolbar id="toolbar" [items]="items"> </ej-toolbar>
 TS:
 public items: Object[] = [{ tooltipText: 'Cut' }]; | **Property** : items[0].tooltipText
 <ejs-toolbar id="toolbar" [items]="items"> </ejs-toolbar>
 TS:
 public items: Object[] = [{ tooltipText: 'Copy' }]; |

| Type | Not Applicable | **Property** : items[0].type
 <ejs-toolbar id="toolbar" [items]="items"> </ej-toolbar>
 TS:
 public items: Object[] = [{type: 'Button'}]; |

| Width | Not Applicable | **Property** : items[0].width
 <ejs-toolbar id="toolbar" [items]="items"> </ej-toolbar>
 TS:
 public items: Object[] = [{width: '50'}]; |

| Disable Items | **Property** : disabledItemIndices
 <ej-toolbar id="toolbar" disabledItemIndices="[]" > </ej-toolbar>| **Method** : enableItems(items, false)
 <ejs-toolbar id="toolbar" #Toolbar ></ej-toolbar>
 TS:
 @ViewChild('Toolbar') public ToolObj: ToolbarComponent;
ToolObj.enableItems([], false); |

| Add Items | Not Applicable | **Method** : addItem(items, index)
 <ejs-toolbar id="toolbar" #Toolbar > </ej-toolbar>
 TS:
 @ViewChild('Toolbar') public ToolObj: ToolbarComponent;
ToolObj.addItem([], 0); |

| Remove Item | **Method** : removeItem(element)
 <ej-toolbar id="toolbar" #Toolbar></ej-toolbar>
 TS:
 @ViewChild('Toolbar') public ToolObj: ToolbarComponent;
ToolObj.removeItem(ele); | **Method** : removeItems(args)
 <ejs-toolbar id="toolbar" #Toolbar > </ej-toolbar>
 TS:
 @ViewChild('Toolbar') public ToolObj: ToolbarComponent;
ToolObj.removeItems(0); |

| RemoveItemById | **Method** : select(index)
 <ej-toolbar id="toolbar" #Toolbar> </ej-toolbar>
 TS:
 @ViewChild('Toolbar') public ToolObj: ToolbarComponent;
ToolObj.select(1); | Not Applicable |

| Enable item | **Method** : enableItem(element)
 <ej-toolbar id="toolbar" #Toolbar> </ej-toolbar>
 TS:
 @ViewChild('Toolbar') public ToolObj: ToolbarComponent;
ToolObj.enableItem(ele); | **Method** : enableItems(items, true)
 <ejs-toolbar id="toolbar" #Toolbar> </ej-toolbar>
 TS:
 @ViewChild('Toolbar') public ToolObj: ToolbarComponent;
ToolObj.enableItems(items, true); |

| EnableItemById | **Method** : enableItemById(id)
 <ej-toolbar id="toolbar" #Toolbar> </ej-toolbar>
 TS:
 @ViewChild('Toolbar') public ToolObj: ToolbarComponent;
ToolObj.enableItemById('left'); | Not Applicable |

| Disable Item | **Method** : disableItem(element)
 <ej-toolbar id="toolbar" #Toolbar> </ej-toolbar>
 TS:
 @ViewChild('Toolbar') public ToolObj: ToolbarComponent;
ToolObj.disableItem(ele); | **Method** : enableItems(items, true)
 <ejs-toolbar id="toolbar" #Toolbar> </ej-toolbar>
 TS:
 @ViewChild('Toolbar') public ToolObj: ToolbarComponent;
ToolObj.enableItems([], false); |

| DisableItemById | **Method** : disableItemById(id)
 <ej-toolbar id="toolbar" #Toolbar> </ej-toolbar>
 TS:
 @ViewChild('Toolbar') public ToolbarObj: ToolbarComponent;
 ToolbarObj.disableItemById('left'); | Not Applicable|

| Show item | Not Applicable| **Method** : hideItem(index, false)
 <ejs-toolbar id="toolbar" #Toolbar> </ejs-toolbar>
 TS:
 @ViewChild('Toolbar') public ToolbarObj: ToolbarComponent;
 ToolbarObj.hideItem(0, false);|

| Hide item | Not Applicable| **Method** : hideItem(index, true)
 <ejs-toolbar id="toolbar" #Toolbar> </ejs-toolbar>
 TS:
 @ViewChild('Toolbar') public ToolbarObj: ToolbarComponent;
 ToolbarObj.hideItem(0, true);|

| Select item | **Method** :selectItem(element)
 <ej-toolbar id="toolbar" #Toolbar> </ej-toolbar>
 TS:
 @ViewChild('Toolbar') public ToolbarObj: ToolbarComponent;
 ToolbarObj.selectItem(ele); | Not Applicable|

| SelectItemById | **Method** : selectItemById(id)
 <ej-toolbar id="toolbar" #Toolbar> </ej-toolbar>
 TS:
 @ViewChild('Toolbar') public ToolbarObj: ToolbarComponent;
 ToolbarObj.selectItemById('left'); | Not Applicable|

| DeselectItemById | **Method** : deselectItemById(id)
 <ej-toolbar id="toolbar" #Toolbar> </ej-toolbar>
 TS:
 @ViewChild('Toolbar') public ToolbarObj: ToolbarComponent;
 ToolbarObj.deselectItemById('left'); | Not Applicable|

| Clicked | Not Applicable| **Event:** clicked
 <ejs-toolbar id="toolbar" (clicked)='onclicked(\$event)'></ej-toolbar>
 TS:
 onclicked(event){ } |

| itemHover | **Event:** itemHover
 <ej-toolbar id='toolbar' (itemHover)='onitemHover(\$event)'></ej-toolbar>
 TS:
 onitemHover(event){ } | Not Applicable|

| itemLeave | **Event:** itemLeave
 <ej-toolbar id='toolbar' (itemLeave)='onitemLeave(\$event)'></ej-toolbar>
 TS:
 onitemHover(event){ } | Not Applicable|

Common

<!-- markdownlint-disable MD033 -->

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

| ----- | ----- | ----- |

| Custom class | **Property** : cssClass
 <ej-toolbar id="toolbar" cssClass="customClass" > </ej-toolbar>| Not Applicable|

| Enabled | **Property** : enabled
 <ej-toolbar id="toolbar" enabled="false"> </ej-toolbar>| Not Applicable|

| EnableSeparator | **Property** : enableSeparator
 <ej-toolbar id="Accordion" [enableSeparator]="false" > </ej-toolbar>| Not Applicable|

| Height | **Property** : height
 <ej-toolbar id="toolbar" height="100%" > </ej-toolbar>| **Property** : height
 <ejs-toolbar id="toolbar" height="100%" > </ejs-toolbar>|

| HtmlAttributes | **Property** : htmlAttributes
 <ej-toolbar id="Accordion" [htmlAttributes]="attributes" > </ej-toolbar>
 TS:
 public attributes: Object = {class: "my-class"}; | Not Applicable |

| Hide | **Property** : hide()
 <ej-toolbar id="Accordion" [hide]="true" > </ej-toolbar> | Not Applicable |

| Orientation | **Property** : orientation
 <ej-toolbar id="Accordion" [orientation]="horizontal" > </ej-toolbar> | Not Applicable |

| OverflowModes | **Property** : responsiveType
 <ej-toolbar id="Accordion" [responsiveType]="popup" > </ej-toolbar> | **Property** : overflowMode
 <ejs-toolbars id="Accordion" [overflowMode]="popup" > </ejs-toolbar> |

| Persistence | Not Applicable | **Property** : enablePersistence
 <ejs-toolbar id="toolbar" [enablePersistence]="false" > </ejs-toolbar> |

| Responsive | **Property** : isResponsive
 <ej-toolbar id="Accordion" [isResponsive]="true" > </ej-toolbar> | Not Applicable |

| ShowRoundedCorner | **Property** : showRoundedCorner
 <ej-toolbar id="toolbar" [showRoundedCorner]="true" > </ej-toolbar> | Not Applicable |

| Width | **Property** : width
 <ej-toolbar id="Accordion" [width]="600" > </ej-toolbar> | **Property** : width
 <ejs-toolbar id="Accordion" [width]="600" > </ejs-toolbar> |

| Enable | **Method** : enable()
 <ej-toolbar id="toolbar" #Toolbar></ej-toolbar>
 TS:
@ViewChild('Toolbar') public ToolObj: ToolbarComponent;
 ToolObj.enable(); | **Method** : disable(false)
 <ejs-toolbar id="toolbar" #Toolbar></ejs-toolbar>
 TS:
@ViewChild('Toolbar') public ToolObj: ToolbarComponent;
 ToolObj.disable(false); |

| Disable | **Method** : disable()
 <ej-toolbar id="toolbar" #Toolbar></ej-toolbar>
 TS:
@ViewChild('Toolbar') public ToolObj: ToolbarComponent;
 ToolObj.disable(); | **Method** : disable(true)
 <ejs-toolbar id="toolbar" #Toolbar></ejs-toolbar>
 TS:
@ViewChild('Toolbar') public ToolObj: ToolbarComponent;
 ToolObj.disable(true); |

| Show | **Method** : show()
 <ej-toolbar id="toolbar" #Toolbar></ej-toolbar>
 TS:
@ViewChild('Toolbar') public ToolObj: ToolbarComponent;
 ToolObj.show(); | Not Applicable |

| Hide | **Method** : hide()
 <ej-toolbar id="toolbar" #Toolbar></ej-toolbar>
 TS:
@ViewChild('Toolbar') public ToolObj: ToolbarComponent;
 ToolObj.hide(); | Not Applicable |

| Refresh | Not Applicable | **Method** : refresh()
 <ejs-toolbar id="toolbar" #Toolbar></ejs-toolbar>
 TS:
@ViewChild('Toolbar') public ToolObj: ToolbarComponent;
 ToolObj.refresh(); |

| Destroy | **Method** : destroy()
 <ej-toolbar id="toolbar" #Toolbar></ej-toolbar>
 TS:
@ViewChild('Toolbar') public ToolObj: ToolbarComponent;
 ToolObj.destroy(); | **Method** : destroy()
 <ejs-toolbar id="toolbar" #Toolbar></ejs-toolbar>
 TS:
@ViewChild('Toolbar') public ToolObj: ToolbarComponent;
 ToolObj.destroy(); |

| BeforeCreate | **Not Applicable** | **Event:** created
 <ejs-toolbar id="toolbar" #Toolbar (created)=‘onbeforeCreate(\$event)’></ejs-toolbar>
 TS:
 onbeforeCreate(event) { } |

| Created | **Event:** create
 <ej-toolbar id="toolbar" #Toolbar (create)=‘oncreate(\$event)’></ej-toolbar>
 TS:
 oncreate(event) { } | **Event:** created
 <ejs-toolbar id="toolbar" #Toolbar (created)=‘oncreated(\$event)’></ejs-toolbar>
 TS:
 oncreated(event) { } |

| Destroyed | **Event:** destroy
 <ej-toolbar id="toolbar" #Toolbar (destroy)=‘ondestroy(\$event)’></ej-toolbar>
 TS:
 ondestroy(event) { } | **Event:** destroyed
 <ejs-toolbar id="toolbar" #Toolbar (destroyed)=‘ondestroyed(\$event)’></ejs-toolbar>
 TS:
 ondestroyed(event) { } |

| FocusOut | **Event:** focusOut
 <ej-toolbar id="toolbar" #Toolbar (focusOut)=‘onfocusOut(\$event)’></ej-toolbar>
 TS:
 onfocusOut(event) { } | **Not Applicable** |

| OverflowOpen | **Event:** overflowOpen
 <ej-toolbar id="toolbar" (overflowOpen)=‘onoverflowOpen(\$event)’></ej-toolbar>
 TS:
 onoverflowOpen(event) { } | **Not Applicable** |

| OverflowClose | **Event:** overflowClose
 <ej-toolbar id="toolbar" (overflowClose)=‘onoverflowClose(\$event)’></ej-toolbar>
 TS:
 onoverflowClose(event) { } | **Not Applicable** |

Tooltip

Getting started with Angular Tooltip component

This section briefly explains how to create a simple **Tooltip** component and configure its available functionalities in angular.

Dependencies

The following list of dependencies are required to use Tooltip component in your application.

```
`javascript
|-- @syncfusion/ej2-angular-popups
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-angular-base
|-- @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.

```
`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 Tooltip 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-popups](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-popups --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-popups@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-popups@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-popups:"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 Tooltip Module

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

```
`javascript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
//Syncfusion ej2-angular-popups module
import { TooltipModule } from '@syncfusion/ej2-angular-popups';
import { AppComponent } from './app.component';
@NgModule({
  imports: [ BrowserModule, TooltipModule ], //declaration of TooltipModule module into NgModule
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Adding CSS Reference

- Add Tooltip component's styles as given below in `styles.css`.

```
[style.css]
`css
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-angular-buttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-angular-popups/styles/material.css";
```

We can also use [CRG](#) to generate combined component styles.

Add Tooltip component

Modify the template in `app.component.ts` file to render the `Tooltip` component. Add the Angular Tooltip by using `<ejs-tooltip>` selector in `template` section of the `app.component.ts` file.

```
`javascript
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  selector: 'my-app',
```

```

template: `
<ejs-tooltip id='tooltip' content='Tooltip content'>
  Hover Me
</ejs-tooltip>`,
encapsulation: ViewEncapsulation.None
})
export class AppComponent { }
`

```

Run the application

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

```

`shell
ng serve --open
`

```

The following code example depicts the way to initialize Tooltip on a single element.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [
    TooltipModule,
    ButtonModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <div id='container' style="display: inline-block; position: relative;
left: 50%;top: 100px;transform: translateX(-50%);">
      <ejs-tooltip id='tooltip' content='Tooltip content'
target="#target">
        <button ejs-button id="target">Show Tooltip</button>
      </ejs-tooltip>
    </div>`,
  encapsulation: ViewEncapsulation.None
})
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));
```

Initialize Tooltip within a container

It is possible to create Tooltip on multiple targets within a container. To do so, define the **selector** property with specific target elements - so that the tooltip will be initialized only on those matched targets within a container. In this case, the Tooltip content gets assigned from the **title** attribute of the target element.

Refer the following code example, to create a Tooltip on multiple targets within a container.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    TooltipModule,
    ButtonModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <div id="tool">
      <ejs-tooltip target='.e-info' position='RightCenter'>
        <form id="details" role="form">
          <table>
            <tr>
              <td class="info">User Name</td>
              <td>
                <input type="text" class="e-info" name="firstname"
title="Please enter your name"> </td>
              </tr>
            <tr>
              <td class="info">Email Address</td>
              <td>
                <input type="text" class="e-info" name="email"
title="Enter a valid email address">
              </td>
            </tr>
            <tr>
              <td class="info">Password</td>
              <td>
                <input type="password" class="e-info"
name="password" title="Be at least 8 characters length">
              </td>
            </tr>
            <tr>
              <td class="info">Confirm Password</td>
              <td>
                <input type="password" class="e-info" name="Cpwd"
title="Re-enter your password">
              </td>
            </tr>
          </table>
        </form>
      </ejs-tooltip>
    </div>
```

```

        </td>
      </tr>
      <tr>
        <td><input ejs-button id="sample" class="center"
type="submit" value="Submit"></td>
        <td><input ejs-button id="clear" class="center"
type="reset" value="Reset"></td>
      </tr>
    </table>
  </form>
</ejs-tooltip>
</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));

```

In the above sample, `details` refers to the container's id, and the target `.e-info` refers to the target elements available

within that container.

See Also

[Positioning Tooltip](#)

[Tooltip Open Mode](#)

[Customize the Tooltip](#)

Schematics in Angular Tooltip component

Angular schematics is a workflow tool that allows you generate component, modules, and resolve dependency issues. The main goal of the schematics is ease of use and development in angular environment.

EJ2 Tooltip supports Angular schematic's module injection, component generation, automation dependency installation, styles imports, etc.

Getting started

Note: Angular schematics supports only from the Angular CLI v6. So, check your version by running `ng --version`. If it is below version 6, update your CLI by running the following command: `npm install -g @angular/cli`.

In order to work with Angular schematics, create an Angular CLI application. Run the following command to create a CLI application.

,

`ng new angular-application`

After running the above command and all the dependency modules installed, we can generate EJ2 Tooltip component using schematics.

Dependency and Module injection using Schematics

Using schematics, we can perform dependency and module injection of the EJ2 Popups package `@syncfusion/ej2-angular-popups` automatically. Run the following command in the root of the application.

```
ng add @syncfusion/ej2-angular-popups --modules=tooltip
```

The above command will do the following,

- Installs the `@syncfusion/ej2-angular-popups` package, and adds an entry in `package.json` file.
- Imports the `TooltipModule` module in the `app.module.ts`, and adds an entry in the `import` property of the `@NgModule` decorator.

Component generation using Schematics

Angular Schematics can be used to generate component, module file, etc. In the same way, we can generate Tooltip components can also be generated.

By using the Schematics to generate EJ2 Tooltip, the time for configuring components is significantly reduced and it is made ready for development immediately. To generate EJ2 Tooltip component with specific features, refer to the following table.

The general syntax for the `ng generate` command is `ng generate @syncfusion/<component-package-name>:<componentName-featureName> --name=<your-desired-name>`

Feature Name	Schematics command
Default Tooltip	<code>ng generate @syncfusion/ej2-angular-popups:tooltip-default --name=default-tooltip</code>
Template Tooltip	<code>ng generate @syncfusion/ej2-angular-popups:tooltip-template --name=template-tooltip</code>

The above commands will do the following,

- Generate the Tooltip with specific features mentioned in the `src/app` with folder name of the `name` property passed.
- Import the generated component into the `app.module.ts` file, and add an entry in the `declarations` array in the `@NgModule` decorator.

Note: It is not required to run the above commands only after the `ng add @syncfusion/ej2-angular-popups`, but it is required to have `@syncfusion/ej2-angular-popups` installed.

Setting dimension in Angular Tooltip component

Height and width

The Tooltip can either be assigned auto height and width values or specific pixel values. The **width** and **height** properties are used to set the outer dimension of the Tooltip element. The default value for both the properties is **auto**. It also accepts string and number values in pixels.

The following sample explains how to set dimensions for the Tooltip.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [

    TooltipModule,
    ButtonModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <div id='container'>
      <ejs-tooltip target='.e-info' position='RightCenter'>
        <form id="details" role="form">
          <table>
            <tr>
              <td class="info">User Name</td>
              <td>
                <input type="text" class="e-info" name="firstname"
title="Please enter your name"> </td>
              </tr>
              <tr>
                <td class="info">Email Address</td>
                <td>
                  <input type="text" class="e-info" name="email"
title="Enter a valid email address">
                </td>
              </tr>
              <tr>
                <td class="info">Password</td>
                <td>
                  <input type="password" class="e-info"
name="password" title="Be at least 8 characters length">
                </td>
              </tr>
              <tr>
                <td class="info">Confirm Password</td>
                <td>
                  <input type="password" class="e-info" name="Cpwd"
title="Re-enter your password">
                </td>
              </tr>
            </tr>
          </table>
        </form>
      </ejs-tooltip>
    </div>
  `
})
```

```

        <td>
            <input id="sample" ej-button class="center"
type="submit" value="Submit">
        </td>
        <td>
            <input id="clear" ej-button type="reset"
class="center" value="Reset" style="align-content: center;">
        </td>
    </tr>
</table>
</form>
</ejs-tooltip>
</div>
`
    encapsulation: ViewEncapsulation.None
  })
  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));

```

Scroll mode

When **height** is specified with a certain pixel value and the Tooltip content overflows, the scrolling mode gets enabled.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [

    TooltipModule,
    ButtonModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <p>A green home is a type of house designed to be
      <ejs-tooltip id='tooltip' width='300px' height='60px'
isSticky='true' [content]='tooltipContent'>
        <a><u>environmentally friendly</u></a>
      </ejs-tooltip> and sustainable. And also focuses on the
efficient use of "energy, water, and building materials." As green homes
      have become more prevalent we have also seen the emergence of
green affordable housing.
    </p>
  `
})

```

```

        encapsulation: ViewEncapsulation.None,
    })
    export class AppComponent {
        tooltipContent: HTMLElement = document.createElement('div');
        constructor() {
            (this as any).tooltipContent =
                '<div id="tooltipContent"><p><b>Environmentally friendly</b> or
environment-friendly, (also referred to as eco-friendly, nature-friendly,
and green) are marketing and sustainability terms referring to goods and
services, laws, guidelines and policies that inflict reduced, minimal, or no
harm upon ecosystems or the environment.</p></div>';
        }
    }
}

```

MAIN.TS

```

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

```

The scrolling mode can best be seen when the sticky mode of the Tooltip is enabled. To enable sticky mode, set the `isSticky` property to true.

Content in Angular Tooltip component

A text or a piece of information assigned to the Tooltip's `content` property will be displayed as the main text stream of the Tooltip.

It can be a string or a template content. If the `content` property is not provided with any specific value, then it takes the value assigned to the `title` attribute of the target element on which the Tooltip was initialized. The content can also dynamically be assigned to the Tooltip via AJAX.

Template content

Any text or image can be added to the Tooltip, by default. To customize the Tooltip layout or to create your own visualized element on the Tooltip, `template` can be used.

Tooltip template content can be rendered using the `ng-template`. If needed it can be rendered using the `HTML` elements also.

The following sample demonstrates how to add content template in tooltip.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    TooltipModule,
    ButtonModule
  ],
  standalone: true,

```

```

    selector: 'my-app',
    template: `
    <span>Using ng-template</span>
    <p>A green home is a type of house designed to be
      <ejs-tooltip id='tooltip_1'>
        <ng-template #content>
          <h3>Template Content</h3>
          <p><strong>Environmentally friendly</strong> or
        <strong>environment-friendly</strong>, <i>(also referred to as eco-friendly,
        nature-friendly, and green)</i> are marketing and sustainability terms
        referring to goods and services, laws, guidelines and policies that inflict
        reduced, minimal, or no harm upon ecosystems or the environment.</p>
        </ng-template>
        <a><u>environmentally friendly</u></a>
      </ejs-tooltip> and sustainable. And also focuses on the efficient
    use of "energy, water, and building materials."
    </p>
    <span>Using HTML Elements</span>
    <p>
      As
      <ejs-tooltip id='tooltip_2' [content]='tooltipContent'>
        <a><u>green homes</u></a>
      </ejs-tooltip>
      have become more prevalent we have also seen the emergence of green
    affordable housing.
    </p>
    `
  })
  export class AppComponent {
    tooltipContent: HTMLElement = document.createElement('div');
    constructor() {
      (this as any).tooltipContent = '<h3>HTML
    Content</h3><p><strong>Environmentally friendly</strong> or
    <strong>environment-friendly</strong>, <i>(also referred to as eco-friendly,
    nature-friendly, and green)</i> are marketing and sustainability terms
    referring to goods and services, laws, guidelines and policies that inflict
    reduced, minimal, or no harm upon ecosystems or the environment.</p>'
    }
  }
}

```

MAIN.TS

```

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

```

Dynamic content via AJAX

The Tooltip content can be dynamically loaded by making use of the AJAX call. The AJAX request is usually made within the `beforeRender` event of the Tooltip, and then the Tooltip's `content` is assigned the value retrieved on it's success.

Note: The Tooltip **target** property includes a unique identifier used to associate Tooltips with specific elements on a webpage or application interface. When setting the Tooltip **target** value as a GUID

(Globally Unique Identifier), it's important to note that the GUID must start with a combination of **letters** before the numeric portion of the GUID. For example, **target: '# + 'tooltip'+ '96ad88bd-294c-47c3-999b-a9daa3285a05'**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { HttpClientModule } from '@angular/common/http'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { Component, ViewChild, Inject } from '@angular/core';
import { TooltipComponent, TooltipEventArgs } from '@syncfusion/ej2-angular-popups';
import { HttpClient, HttpClientModule } from '@angular/common/http';
@Component({
  imports: [

    HttpClientModule,
    TooltipModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <div id="tool">
      <h4>National Sports</h4>
      <ejs-tooltip #tooltip id="tooltip" class="e-prevent-select"
content='Loading...' target="#countryList [title]"
position='RightCenter' (beforeRender)="onBeforeRender($event)">
        <div id="countryList">
          <ul>
            <li class="target" title="1"><span>Australia</span></li>
            <li class="target" title="2"><span>Bhutan</span></li>
            <li class="target" title="3"><span>China</span></li>
            <li class="target" title="4"><span>Cuba</span></li>
            <li class="target" title="5"><span>India</span></li>
            <li class="target" title="6"><span>Switzerland</span></li>
            <li class="target" title="7"><span>United
States</span></li>
          </ul>
        </div>
      </ejs-tooltip>
    </div>
  `,
  styles: [`
    #countryList {
      padding: 5px;
    }
    #countryList ul {
      list-style-type: none;
      margin: 0;
      padding: 0;
      width: 100px;
      border: 1px solid #c4c4c4;
    }
    #countryList li {
      padding: 10px;
    }
  `]
})
```



```

    }
    #countryList li:hover {
        background-color: #ececec;
    }
    .contentWrap {
        padding: 3px 0;
        line-height: 16px;
    }
    .def {
        float: right;
    }
    #tool {
        width: 350px;
        position: relative;
        left: 50%;
        transform: translateX(-25%);
    }
    `]
})
export class AppComponent {
    @ViewChild('tooltip')
    public tooltipControl: TooltipComponent | any ;
    constructor( @Inject(HttpClientModule) public http: HttpClient) { }
    onBeforeRender(args: TooltipEventArgs) {
        (this as any).http.get('tooltipdata.json')
            .map((res: { json: () => any; }) => res.json())
            .subscribe(
                (result: any) => {
                    for (let i: number = 0; i < result.length; i++) {
                        if (result[i].Id === args.target.getAttribute('data-
content')) {
                            this.tooltipControl!.content = "<div
class='contentWrap'><div class='def'>" + result[i].Sports + "</div></div>";
                        }
                    }
                    this.tooltipControl!.dataBind();
                },
                (err: Response) => {
                    this.tooltipControl!.content = err.statusText;
                    this.tooltipControl!.dataBind();
                }
            );
    }
}

```

MAIN.TS

```

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

```

Position in Angular Tooltip component

Tooltips can be attached to 12 static locations around the target.

On initializing the Tooltip, you can set the position property with any one of the following values:

- TopLeft
- TopCenter
- TopRight
- BottomLeft
- BottomCenter
- BottomRight
- LeftTop
- LeftCenter
- LeftBottom
- RightTop
- RightCenter
- RightBottom

By default, Tooltip is placed at the **TopCenter** of the target element.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewChild, ViewEncapsulation } from '@angular/core';
import { TooltipComponent, Position } from '@syncfusion/ej2-angular-popups';
@Component({
  imports: [

    TooltipModule,
    ButtonModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <div style='display: inline-flex;position: relative;left:
50%;transform: translateX(-50%);margin-top: 100px;'>
      <div>
        <ejs-tooltip #tooltip id='tooltip' content='Tooltip content' >
          <button ejs-button id="target">Show Tooltip</button>
        </ejs-tooltip>
      </div>
      <div>
        <select id="positions" (change)="onChange($event.target)"
class="form-control">
          <option value="TopLeft">Top Left</option>
          <option value="TopCenter" selected>Top Center</option>
          <option value="TopRight">Top Right</option>
          <option value="BottomLeft">Bottom Left</option>
          <option value="BottomCenter">Bottom Center</option>
          <option value="BottomRight">Bottom Right</option>
          <option value="LeftTop">Left Top</option>
          <option value="LeftCenter">Left Center</option>
          <option value="LeftBottom">Left Bottom</option>
          <option value="RightTop">Right Top</option>
          <option value="RightCenter">Right Center</option>
```

```

        <option value="RightBottom">Right Bottom</option>
      </select>
    </div>
  </div>
  ,
  styles: [
    ,
    #tooltip {
      padding: 5px;
      margin-right: 20px;
    }
    ,
  ],
  encapsulation: ViewEncapsulation.None,
})
export class AppComponent {
  @ViewChild('tooltip') public control : TooltipComponent | any;
  onChange(value: string | any) {
    this.control.position = value as Position;
  }
}

```

MAIN.TS

```

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

```

Tip pointer positioning

The Tooltip pointer can be attached or detached from the Tooltip by using the `showTipPointer` property.

Pointer positions can be adjusted using the `tipPointerPosition` property that can be assigned to one of the following values:

- Auto
- Start
- Middle
- End

The following code example illustrates how to set the pointer to the start position of the Tooltip.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TooltipModule } from '@syncfusion/ej2-angular-popups';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { Component } from '@angular/core';
@Component({
  imports: [

```

```

        TooltipModule,
        ButtonModule
    ],
    standalone: true,
    selector: 'my-app',
    template: `
        <ejs-tooltip id="tooltip" content='Tooltip content'
        tipPointerPosition='Start'>
            <button ejs-button>Show tooltip</button>
        </ejs-tooltip>
    `,
    styles: [
        #tooltip {
            display: block;
            margin: 80px auto;
            width: 200px;
            text-align: center;
            color: #424242;
            font-size: 20px;
        }
    ]
})
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));

```

By default, tip pointers are auto adjusted so that the arrow does not point outside the target element.

Dynamic positioning

The Tooltip and its tip pointer can be positioned dynamically based on the target's location. This can be achieved by using the `refresh` method, which auto adjusts the Tooltip over the target.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewChild, Inject } from '@angular/core';
import { TooltipComponent } from '@syncfusion/ej2-angular-popups';
import { Draggable } from '@syncfusion/ej2-base';
@Component({
    imports: [

        TooltipModule,
        ButtonModule
    ],
    standalone: true,
    selector: 'my-app',
    template: `

```

```

    <ejs-tooltip #tooltip id='targetContainer' content='Drag me !!!'
    target='#demoSmart' [animation]='tooltipAnimation'>
      <div id="demoSmart">
        </div>
      </ejs-tooltip>
    `
    styles: [
      #targetContainer {
        position: relative;
        height: 360px;
        overflow: hidden;
        border: 1px solid #c8c8c8;
        display: block;
      }
      #demoSmart {
        width: 50px;
        height: 50px;
        background: rebeccapurple;
        position: absolute;
        left: 35%;
        top: 35%;
      }
    ]
  })
  export class AppComponent {
    @ViewChild('tooltip')
    public tooltipControl: TooltipComponent | any ;
    public tooltipAnimation: Object = {
      open: { effect: 'None' },
      close: { effect: 'None' }
    };
    ngOnInit(): void {
      let ele: HTMLElement = document.getElementById('demoSmart') as
      HTMLElement;
      let drag: Draggable = new Draggable(ele, {
        clone: false,
        dragArea: '#targetContainer',
        drag: (args: any) => {
          this.tooltipControl.refresh(args.element);
        },
        dragStart: (args: any) => {
          this.tooltipControl.open(args.element);
        },
        dragStop: (args: any) => {
          this.tooltipControl.close();
        }
      });
    }
  }
}

```

MAIN.TS

```

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

```

Mouse trailing

Tooltips can be positioned relative to the mouse pointer. This behavior can be enabled or disabled by using the `mouseTrail` property.

By default, it is set to `false`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    TooltipModule,
    ButtonModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <ejs-tooltip id="tooltip" content='Tooltip content'
[mouseTrail]='true'>
      <span>Show tooltip</span>
    </ejs-tooltip>
  `,
  styles: [
    `#tooltip {
      display: block;
      background-color: #cfd8dc;
      border: 3px solid #eceff1;
      box-sizing: border-box;
      margin: 80px auto;
      padding: 20px 0;
      width: 200px;
      text-align: center;
      color: #424242;
      font-size: 20px;
    }`
  ]
})
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));
```

When mouse trailing option is enabled, the tip pointer position gets auto adjusted based on the target, and

other position values like start, end, and middle are not applied (to prevent the pointer from moving out of target).

Setting offset values

Offset values are set to specify the distance between the target and tooltip element.

`offsetX` and `offsetY` properties are used to specify the offset left and top values.

- `offsetX` specifies the distance between the target and Tooltip element in X axis.
- `offsetY` specifies the distance between the target and Tooltip element in Y axis.

The following code example illustrates how to set offset values.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [

    TooltipModule,
    ButtonModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <ejs-tooltip id="tooltip" content='Tooltip content' [offsetX]='30'
[offsetY]='30' [mouseTrail]='true' [showTipPointer]='false'>
      <span>Show tooltip</span>
    </ejs-tooltip>
  `,
  styles: [`
    #tooltip {
      display: block;
      background-color: #cfd8dc;
      border: 3px solid #eceff1;
      box-sizing: border-box;
      margin: 80px auto;
      padding: 20px 0;
      width: 200px;
      text-align: center;
      color: #424242;
      font-size: 20px;
    }
  `]
})
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));
```

By default, collision is handled automatically and therefore when collision is detected the Tooltip fits horizontally and flips vertically.

Open mode in Angular Tooltip component

You can decide the mode on which the Tooltip is to be opened on a page, i.e., on hovering, focusing, or clicking on the target elements.

On mobile devices, Tooltips appear when you tap and hold the element, even if the `opensOn` option is assigned with `Hover`.

Tooltips are also displayed as long as you continue to tap and hold the element. On lift, it disappears in the next 1.5 seconds.

If there is another action before that time ends, then the Tooltip disappears.

The `opensOn` property can take either a single or a combination of multiple values, separated by space from the following options.

The table below explains how the Tooltip opens on both desktop and mobile based on the value(s) assigned to the `opensOn` property.

By default, it takes `Auto` value.

Values	Desktop	Mobile
<code>Auto</code>	Tooltip appears when you hover over the target or when the target element receives the focus.	Tooltip opens on tap and hold of the target element.
<code>Hover</code>	Tooltip appears when you hover over the target.	Tooltip opens on tap and hold of the target element.
<code>Click</code>	Tooltip appears when you click a target element.	Tooltip appears when you single tap the target element.
<code>Focus</code>	Tooltip appears when you focus (say through tab key) on a target element.	Tooltip appears with a single tap on the target element.
<code>Custom</code>	Tooltip is not triggered by any default action. So, you have to bind your own events and use either <code>open</code> or <code>close</code> public methods. Same as Desktop.	

To open the Tooltip for multiple actions, say while hovering over or clicking on a target element, the `opensOn` property can be assigned

with multiple values, separated by space as `hover click`.

`Auto` value cannot be used with any combination for multiple values.

The following code example shows how to set the open mode for Tooltips.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewChild } from '@angular/core';
import { TooltipComponent } from '@syncfusion/ej2-angular-popups';
@Component({
  imports: [

    TooltipModule,
    ButtonModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <div id='container'>
      <table style="margin: 150px auto 0 auto;transform: translateY(-
50%);">
        <tbody>
          <tr>
            <td style="padding:25px">
              <ejs-tooltip id="tooltipHover" opensOn='Hover'
content='Tooltip from hover' >
                <div>
                  <input ej2-button class="blocks" type="button"
value="Hover Me!"/></div>
              </ejs-tooltip>
            </td>
            <td style="padding:25px">
              <ejs-tooltip id="tooltipClick" opensOn='Click'
content='Tooltip from click'>
                <div>
                  <input ej2-button class="blocks" type="button"
value="Click Me!"/></div>
              </ejs-tooltip>
            </td>
          </tr>
          <tr>
            <td style="padding:25px">
              <ejs-tooltip id="tooltipFocus" opensOn='Focus'
content='Tooltip from focus' target='#input'>
                <div>
                  <input id='input' class="blocks e-float-input"
type="text" placeholder="Focus and blur"/>
                </div>
              </ejs-tooltip>
            </td>
            <td style="padding:25px">
              <ejs-tooltip id="tooltipCustom" #tooltipCustom
opensOn='custom' content='Tooltip from custom mode'>
                <div>
                  <input ej2-button class="blocks" id="tooltipOpen"
type="button" value="Click to open tooltip manually"
(click)='onCustomOpen($event)' />
                </div>
            </td>
          </tr>
        </tbody>
      </table>
    </div>
  `
})

```

```

        </ejs-tooltip>
      </td>
    </tr>
  </tbody>
</table>
</div>
`
,
styles: [
  .blocks {
    width: 260px;
  }
]
})
export class AppComponent {
  @ViewChild('tooltipCustom')
  public tooltipCustom : TooltipComponent | any;
  constructor() { }
  onCustomOpen(args: any): void {
    if (args.target.getAttribute('data-tooltip-id')) {
      this.tooltipCustom.close();
    } else {
      this.tooltipCustom.open(args.target);
    }
  }
}
}

```

MAIN.TS

```

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

```

Custom open mode

Other than the above specified options, the **custom** mode makes the Tooltip appear on screen for user-defined custom actions such as **right-click**, **double-click**, and so on. Here, the tooltip is not triggered by any default action, and you have to bind your own events and use either **open** or **close** public methods to show or hide the Tooltips.

The following code example shows how to define custom open mode for the Tooltip.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewChild } from '@angular/core';
import { TooltipComponent } from '@syncfusion/ej2-angular-popups';
@Component({
  imports: [
    TooltipModule,
    ButtonModule
  ],

```

```

standalone: true,
  selector: 'my-app',
  template: `
    <ejs-tooltip #tooltip id="tooltip" content='Tooltip from custom mode'
    opensOn='Custom' (dblclick)=customOpen($event) '>
      <span>Double-click to open Tooltip</span>
    </ejs-tooltip>
  `,
  styles: [
    `#tooltip {
      display: block;
      background-color: #cfd8dc;
      border: 3px solid #eceff1;
      box-sizing: border-box;
      margin: 80px auto;
      padding: 20px 0;
      width: 200px;
      text-align: center;
      color: #424242;
      font-size: 20px;
    }`
  ]
})
export class AppComponent {
  @ViewChild('tooltip')
  public tooltipControl: TooltipComponent | any;
  constructor() {}
  customOpen(args: any): void {
    if (args.target.getAttribute("data-tooltip-id")) {
      this.tooltipControl.close();
    } else {
      this.tooltipControl.open(args.target);
    }
  }
}

```

MAIN.TS

```

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

```

Sticky mode

With this mode set to `true`, Tooltips can be made to show up on the screen as long as the close icon is pressed. In this mode, close icon is attached to the Tooltip located at the top right corner. This mode can be enabled or disabled using the `isSticky` property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';

```

```
@Component({
  imports: [
    TooltipModule,
    ButtonModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <ejs-tooltip id="tooltip" content='Click close icon to close me'
    [isSticky]='true'>
      <span>Show tooltip</span>
    </ejs-tooltip>
  `,
  styles: [`
    #tooltip {
      display: block;
      background-color: #cfd8dc;
      border: 3px solid #eceff1;
      box-sizing: border-box;
      margin: 80px auto;
      padding: 20px 0;
      width: 200px;
      text-align: center;
      color: #424242;
      font-size: 20px;
    }
  `]
})
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));
```

Open/Close Tooltip with delay

The Tooltips can be opened or closed after some delay by using the `openDelay` and `closeDelay` properties.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    TooltipModule,
    ButtonModule
```

```

    ],
    standalone: true,
    selector: 'my-app',
    template: `
      <ejs-tooltip id="tooltip" content='Tooltip with delay'
        [openDelay]='1000' [closeDelay]='1000'>
        Show tooltip
      </ejs-tooltip>
    `,
    styles: [
      `
      #tooltip {
        display: block;
        background-color: #cfd8dc;
        border: 3px solid #eceff1;
        box-sizing: border-box;
        margin: 80px auto;
        padding: 20px 0;
        width: 200px;
        text-align: center;
        color: #424242;
        font-size: 20px;
      }
    `
    ]
  })
  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));

```

Animation in Angular Tooltip component

To animate the Tooltip, a set of specific animation effects are available, and it can be controlled using the `animation` property.

The `animation` property also allows you to set delay, duration, and various other effects of your choice.

[AnimationModel](#) is derived from base to apply the chosen animation effect, duration, etc. on Tooltips.

By default, Tooltip entrance occurs over 150 ms using the `ease-out` timing function. It exits also at 150 ms, but uses `ease-in` timing function.

The default animation effect for the Tooltip is set to `FadeIn` for its open action, and `FadeOut` for its close action.

The default `duration` is set to 150 ms and `delay` is set to 0.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TooltipModule } from '@syncfusion/ej2-angular-popups';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';

```

```

import { Component } from '@angular/core';
@Component({
  imports: [
    TooltipModule,
    ButtonModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <ejs-tooltip id="tooltip" content='Tooltip animation effect'
[animation]='TooltipAnimation'>
      Show tooltip
    </ejs-tooltip>
  `,
  styles: [ `
#tooltip {
  display: block;
  background-color: #cfd8dc;
  border: 3px solid #eceff1;
  box-sizing: border-box;
  margin: 80px auto;
  padding: 20px 0;
  width: 200px;
  text-align: center;
  color: #424242;
  font-size: 20px;
}
` ]
})
export class AppComponent {
  public TooltipAnimation: Object = {
    open: { effect: 'ZoomIn', duration: 1000, delay: 0 },
    close: { effect: 'ZoomOut', duration: 500, delay: 0 }
  };
}

```

MAIN.TS

```

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

```

Animation effects

The animation effects that are applicable to Tooltips are:

- FadeIn
- FadeOut
- FadeZoomIn
- FadeZoomOut
- FlipXDownIn
- FlipXDownOut

- FlipXUpIn
- FlipXUpOut
- FlipYLeftIn
- FlipYLeftOut
- FlipYRightIn
- FlipYRightOut
- ZoomIn
- ZoomOut
- None

When the `effect` is specified as `none`, no effect will be applied to the Tooltip, and animation is considered to be set to `off`.

Some of the above animation effects suits the Tooltip better when its tip pointer is hidden.

This can be achieved by setting the `showTipPointer` to false.

Animating via open/close methods

Animations can also be applied on Tooltips dynamically through `open` and `close` methods. These methods accept the animation model as an optional parameter. If you pass `TooltipAnimationSettings`, animation takes this model; otherwise, it takes the values from the `animation` property. It is also possible to pass different animations for Tooltips on each target.

Refer to the code snippet below to apply animations using public methods.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewChild } from '@angular/core';
import { TooltipComponent, TooltipAnimationSettings } from '@syncfusion/ej2-angular-popups';
@Component({
  imports: [

    TooltipModule,
    ButtonModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <ejs-tooltip id="tooltip" #tooltipAnimate content='Tooltip animation
    effect' opensOn='Custom' (click)='onCustomClick($event)'>
      Show tooltip
    </ejs-tooltip>
  `,
  styles: [`
    #tooltip {
      display: block;
      background-color: #cfd8dc;
      border: 3px solid #eceff1;
      box-sizing: border-box;
      margin: 80px auto;
    }
  `]
})
```

```

        padding: 20px 0;
        width: 200px;
        text-align: center;
        color: #424242;
        font-size: 20px;
    }
    `]
  })
  export class AppComponent {
    @ViewChild('tooltipAnimate')
    public tooltipControl : TooltipComponent | any;
    onCustomClick(args: any): void {
      if (args.target.getAttribute('data-tooltip-id')) {
        let closeAnimation: TooltipAnimationSettings = { effect:
'FadeOut', duration: 1000 }
        this.tooltipControl.close(closeAnimation);
      } else {
        let openAnimation: TooltipAnimationSettings = { effect:
'FadeIn', duration: 1000 }
        this.tooltipControl.open(args.target, openAnimation);
      }
    }
  }
}

```

MAIN.TS

```

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

```

Apply transition

The transition effect can be applied on Tooltips by using the `beforeOpen` event as given in the following work-around code example.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewEncapsulation, ViewChild } from '@angular/core';
import { TooltipComponent, TooltipEventArgs } from '@syncfusion/ej2-angular-
popups';
@Component({
  imports: [

    TooltipModule,
    ButtonModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <h3> Transition effect </h3>

```



```

    <ejs-tooltip id="tooltip" #tooltip class="e-prevent-select"
    target='.circle-tool' [closeDelay]='1000' [animation]='Animation'
    (beforeRender)='onBeforeRender($event)'
    (beforeOpen)='onBeforeOpen($event)' (afterClose)='onAfterClose($event)'>
      <div class="circle-tool" style="top:18%;left:5%" title="This is
    Turtle !!!"></div>
      <div class="circle-tool" style="top:30%;left:30%" title="This is
    Snake !!!"></div>
      <div class="circle-tool" style="top:80%;left:80%" title="This is
    Croc !!!"></div>
      <div class="circle-tool" style="top:65%;left:50%" title="This is
    String Ray !!!"></div>
      <div class="circle-tool" style="top:75%;left:15%" title="This is
    Blob Fish !!!"></div>
      <div class="circle-tool" style="top:30%;left:70%" title="This is
    Mammoth !!!"></div>
    </ejs-tooltip>
    `,
    styles: [`
    #tooltip {
      display: block;
      border: 1px solid #c8c8c8;
      height: 200px;
      margin-left: 10px;
      margin-right: 10px;
      position: relative;
    }
    .circle-tool {
      position: absolute;
      width: 20px;
      height: 20px;
      background: #9acd32;
      -moz-border-radius: 50px;
      -webkit-border-radius: 50px;
      border-radius: 50px;
    }
    `],
    encapsulation: ViewEncapsulation.None
  })
}
export class AppComponent {
  @ViewChild('tooltip')
  public tooltipControl: TooltipComponent | any ;
  public Animation: Object = {
    open: { effect: 'ZoomIn', duration: 500 },
    close: { effect: 'ZoomOut', duration: 500 }
  };
  onBeforeRender(args: TooltipEventArgs): void {
    if (args.element) {
      this.tooltipControl.animation = { open: { effect: 'None' } };
    }
  }
  onBeforeOpen(args: TooltipEventArgs): void {
    if (args.element) {
      args.element.style.display = 'block';
      args.element.style.transitionProperty = 'left,top';
      args.element.style.transitionDuration = '1000ms';
    }
  }
}

```

```

    }
    onAfterClose(args: TooltipEventArgs): void {
        this.tooltipControl.animation = {
            open: { effect: 'ZoomIn', duration: 500 },
            close: { effect: 'ZoomOut', duration: 500 }
        };
    }
}

```

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 Tooltip component

The Tooltip can be customized by using the `cssClass` property, which accepts custom CSS class names that define specific user-defined styles and themes to be applied on the Tooltip element.

Tip pointer customization

Styling the tip pointer's size, background, and border colors can be done using the `cssClass` property, as given below.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TooltipModule } from '@syncfusion/ej2-angular-popups';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
    imports: [
        TooltipModule,
        ButtonModule
    ],
    standalone: true,
    selector: 'my-app',
    template: `
        <ejs-tooltip id="tooltip" content='Tooltip arrow customized'
        cssClass='custom-tip'>
            Show tooltip
        </ejs-tooltip>
    `,
    styles: [
        `#tooltip {
            display: block;
            background-color: #cfd8dc;
            border: 3px solid #eceff1;
            box-sizing: border-box;
            margin: 80px auto;
            padding: 20px 0;
            width: 200px;
            text-align: center;
        }`
    ]
})

```

```
        color: #424242;
        font-size: 20px;
    }
    .custom-tip.e-tooltip-wrap {
        padding: 4px;
    }
    .custom-tip.e-tooltip-wrap .e-arrow-tip.e-tip-bottom {
        height: 20px;
        width: 12px;
    }
    .custom-tip.e-tooltip-wrap .e-arrow-tip.e-tip-top {
        height: 20px;
        width: 12px;
    }
    .custom-tip.e-tooltip-wrap .e-arrow-tip.e-tip-left {
        height: 12px;
        width: 20px;
    }
    .custom-tip.e-tooltip-wrap .e-arrow-tip.e-tip-right {
        height: 12px;
        width: 20px;
    }
    .custom-tip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
        border-left: 6px solid transparent;
        border-right: 6px solid transparent;
        border-top: 20px solid #616161;
    }
    .custom-tip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
        border-bottom: 20px solid #616161;
        border-left: 6px solid transparent;
        border-right: 6px solid transparent;
    }
    .custom-tip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-left {
        border-bottom: 6px solid transparent;
        border-right: 20px solid #616161;
        border-top: 6px solid transparent;
    }
    .custom-tip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-right {
        border-bottom: 6px solid transparent;
        border-left: 20px solid #616161;
        border-top: 6px solid transparent;
    }
    .custom-tip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom {
        border-left: 12px solid transparent;
        border-right: 5px solid transparent;
        border-top: 19px solid #616161;
    }
    .custom-tip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-top {
        border-bottom: 19px solid #616161;
        border-left: 5px solid transparent;
        border-right: 5px solid transparent;
    }
    .custom-tip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-left {
        border-bottom: 5px solid transparent;
        border-right: 19px solid #616161;
        border-top: 5px solid transparent;
    }
}
```

```

        .custom-tip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-right {
            border-bottom: 5px solid transparent;
            border-left: 19px solid #616161;
            border-top: 5px solid transparent;
        }
    `],
    encapsulation: ViewEncapsulation.None
  })
  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));

```

Tooltip customization

The complete look and feel of the Tooltip can be customized by changing its background color, opacity, content font, etc.

The following code example shows the way to achieve it.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [
    TooltipModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <ejs-tooltip id="tooltip" content='Tooltip customized'
    cssClass='customtooltip'>
      Show tooltip
    </ejs-tooltip>
  `,
  styleUrls: ['./custom.css'],
  encapsulation: ViewEncapsulation.None
})
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 Tooltip 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 tooltip

Use the following CSS to customize the tooltip.

```
`css
.e-tooltip-wrap {
border-radius: 4px;
opacity: 1;
}
```

Customizing the tooltip popup

Use the following CSS to customize the tooltip popup properties.

```
`css
.e-tooltip-wrap.e-popup {
background-color: #fff;
border: 2px solid #000;
}
```

Customizing the tooltip content

Use the following CSS to customize the tooltip content.

```
`css
.e-tooltip-wrap .e-tip-content {
color: red;
font-size: 12px;
line-height: 20px;
}
```

Customizing the tooltip arrow tip

Use the following CSS to customize the tooltip arrow tip.

```
`css
/ To customize the arrow tip at bottom /
.e-tooltip-wrap .e-arrow-tip.e-tip-bottom {
height: 12px;
```

```
left: 50%;
top: 100%;
width: 24px;
}
/ To customize the arrow tip at top /
.e-tooltip-wrap .e-arrow-tip.e-tip-top {
height: 12px;
left: 50%;
top: -9px;
width: 24px;
}
/ To customize the arrow tip at left /
.e-tooltip-wrap .e-arrow-tip.e-tip-left {
height: 24px;
left: -9px;
top: 48%;
width: 12px;
}
/ To customize the arrow tip at right /
.e-tooltip-wrap .e-arrow-tip.e-tip-right {
height: 24px;
left: 100%;
top: 50%;
width: 12px;
}
`
```

[Customizing the tooltip inner tip](#)

Use the following CSS to customize the tooltip inner tip.

```
`css
.e-tooltip-wrap .e-arrow-tip-inner.e-tip-right,
.e-tooltip-wrap .e-arrow-tip-inner.e-tip-left,
.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom,
.e-tooltip-wrap .e-arrow-tip-inner.e-tip-top {
```

```
color: #fff;
font-size: 25.9px;
}
`
```

Customizing the tooltip outer tip

Use the following CSS to customize the tooltip outer tip.

```
`css
/ To customize the arrow tip at bottom /
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
border-left: 12px solid transparent;
border-right: 14px solid transparent;
border-top: 12px solid #000;
}
/ To customize the arrow tip at top /
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
border-bottom: 12px solid #000;
border-left: 12px solid transparent;
border-right: 12px solid transparent;
}
/ To customize the arrow tip at left /
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-left {
border-bottom: 12px solid transparent;
border-right: 12px solid #000;
border-top: 12px solid transparent;
}
/ To customize the arrow tip at right /
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-right {
border-bottom: 12px solid transparent;
border-left: 12px solid #000;
border-top: 12px solid transparent;
}
`
```

Accessibility in Angular Tooltip component

The Tooltip 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 Tooltip 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 Tooltip component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Tooltip component.

Attributes	Description
role="tooltip"	The element that serves as the container for the tooltip has the ARIA role of <code>tooltip</code> .
aria-describedby	This attribute is added to the target element on which the Tooltip gets opened, when focusing or hovering over it. It usually holds the randomly generated <code>Id</code> value of the Tooltip element. In case, the target element already holds an <code>aria-describedby</code> attribute with <code>Id</code> value of some other component, then the Tooltip <code>Id</code> value can be additionally appended to the existing <code>aria-describedby</code> attribute separated by a space as shown in the example below. example: <code>aria-describedby = "my-text my-tooltip"</code> <code>my-text</code> is the <code>Id</code> of some other component. <code>my-tooltip</code> is the <code>id</code> of Tooltip component. When the Tooltip is closed, the <code>aria-describedby</code> attribute is removed from the target.
aria-hidden	This attribute is assigned to the Tooltip element whose default value is <code>true</code> . When <code>true</code> , it denotes that the Tooltip element is in a hidden or a closed state. When the Tooltip appears on the screen, its value changes to <code>false</code> .

Keyboard interaction

The Tooltip 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 Tooltip component.

Keys	Description
Escape	Closes or dismisses the Tooltip.
Tab	A form control receiving focus (say through tab key), opens the Tooltip, and on focus out closes it.

1. When the Tooltip is being displayed on the target element, focus continues to stay on it.
2. If the Tooltip opens on mouse entering into the target element space, then it should be dismissed only when the mouse leaves that target.
3. If the Tooltip opens on the target element that receives focus, then it should be closed only when the focus moves out of that target element.

Likewise, if the Tooltip opens on a click, then it should be closed only on another click action.

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion Angular components](#)

Ej1 api migration in Angular Tooltip component

This article describes the API migration process of Tooltip component from Essential JS 1 to Essential JS 2

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Position | **Property:** *position* `
<ej-tooltip content='Tooltip' [position]='target'>Tooltip</ej-tooltip>

public target = {
 target: {
 horizontal: 'center',
 vertical: 'top'
 },
 stem: {
 horizontal: 'center',
 vertical: 'bottom'
 } } | Property: position
<ejs-tooltip position="TopCenter" content='Tooltip'>Tooltip</ejs-tooltip> |`

| Animation | **Property:** *animation* `
<ej-tooltip content='Tooltip' [animation]='animation'>Tooltip</ej-tooltip>

public animation = { effect : "slide", speed : 1000}; | Property: animation
<ejs-tooltip [animation]="animation" content='Tooltip'>Tooltip</ejs-tooltip>

public animation = { open: { effect: 'FadeIn', duration: 150, delay: 0 }, close: { effect: 'FadeOut', duration: 150, delay: 0 } } |`

| Close Time Out | **Property:** *autoCloseTimeout* `
<ej-tooltip content='Tooltip' autoCloseTimeout=5000>Tooltip</ej-tooltip> | Property: closeDelay, openDelay
 <ejs-tooltip [openDelay]="500" [closeDelay]="500" content='Tooltip'>Tooltip</ejs-tooltip> |`

| Sticky Mode | **Property:** *closeMode* `
<ej-tooltip content='Tooltip' closeMode='sticky'>Tooltip</ej-tooltip> | Property: isSticky
<ejs-tooltip [isSticky]="true" content='Tooltip'>Tooltip</ejs-tooltip> |`

| Offset from target | **Property:** *tip.adjust.xValue/ tip.adjust.yValue* `
<ej-tooltip content='Tooltip' [tip]='tip'>Tooltip</ej-tooltip>

public tip = { size : { width : 25, height : 12}, adjust : {xValue : 5, yValue: 6}}; | Property: offsetX/ offsetY
 <ejs-tooltip [offsetX]="10" [offsetY]="10" content='Tooltip'>Tooltip</ejs-tooltip> |`

| Mouse trail on target | **Property:** *associate* `
<ej-tooltip content='Tooltip' associate='mouse'>Tooltip</ej-tooltip> | Property: mouseTrail
<ejs-tooltip [mouseTrail]="true" content='Tooltip'>Tooltip</ejs-tooltip> |`

| Open mode of tooltip | **Not Applicable** | **Property:** *opensOn* `
<ejs-tooltip [opensOn]="Click" content='Tooltip'>Tooltip</ejs-tooltip> |`

| Enable disable the tip of tooltip | **Property:** *isBalloon* `
<ej-tooltip content='Tooltip' [isBalloon]='false'>Tooltip</ej-tooltip> | Property: showTipPointer
<ejs-tooltip [showTipPointer]="false" content='Tooltip'>Tooltip</ejs-tooltip> |`

| Hide | **Method:** *hide()* `
<ej-tooltip #tooltipInstance content='Tooltip'>Tooltip</ej-tooltip>

this.tooltipInstance.widget.hide(); | Method: close()
<ejs-tooltip #tooltipInstance content='Tooltip'>Tooltip</ejs-tooltip>

</>this.tooltipInstance.close(); |`

```
| Show | Method: show() <br /> <ej-tooltip #tooltipInstance content='Tooltip'>Tooltip</ej-  
tooltip><br /><br />this.tooltipInstance.widget.show();| Method: open() <br /><ejs-tooltip  
#tooltipInstance content='Tooltip'>Tooltip</ejs-tooltip><br /><br />  
>this.tooltipInstance.open();|  
  
| Close | Event: close <br /><ej-tooltip content='Tooltip' (close)="close">Tooltip</e-tooltip><br  
><br />public close() { }; | Event: afterClose <br /><ejs-tooltip content='Tooltip'  
(afterClose)="onAfterClose">Tooltip</ejs-tooltip><br /><br />public onAfterClose() { };|  
  
| Open | Event: open <br /> <ej-tooltip content='Tooltip' (open)="open ">Tooltip</e-  
tooltip><br /><br />public open () { }; | Event: afterOpen <br /> <ejs-tooltip content='Tooltip'  
(onAfterOpen)="afterOpen">Tooltip</ejs-tooltip><br /><br />public onAfterOpen() { };  
  
| Before Collision | Not Applicable | Event: beforeCollision <br /><ejs-tooltip content='Tooltip'  
(onBeforeCollision)="beforeCollision">Tooltip</ejs-tooltip><br /><br />public  
onBeforeCollision() { };|  
  
| Tracking | Event: tracking <br /><ej-tooltip content='Tooltip' (tracking)="tracking ">Tooltip</e-  
tooltip><br /><br />public tracking () { }; | Method: open(),close(),refresh() <br /><ejs-tooltip  
#tooltip id="targetContainer" #tooltip content="Drag me anywhere, to start walking with me  
!!!" [offsetX]="-15" target="#demoSmart" [animation]="tooltipAnimation"><br /><div  
id="demoSmart"> </div><br /></ejs-tooltip><br /><br />let ele: HTMLElement =  
document.getElementById('demoSmart');<br />let <br />drag: Draggable = new Draggable(ele,  
{<br />clone: false,<br />dragArea: '#targetContainer',<br />drag: (args: any) => {<br />if  
(args.element.getAttribute('data-tooltip-id') === null) {<br />tooltip.open(args.element);<br />}  
else {<br />tooltip.refresh(args.element);<br />}<br />,<br />dragStart: (args: any) => {<br />if  
(args.element.getAttribute('data-tooltip-id') !== null) { return; }<br />  
>tooltip.open(args.element);<br />,<br />dragStop: (args: any) => {<br />tooltip.close();<br />  
>}<br />}); |
```

How To

Show tooltip on disabled elements and disable tooltip in Angular Tooltip component

By default, Tooltips will not be displayed on disabled elements. However, it is possible to enable this behavior by following the steps below.

1. Add a disabled element like the `button` element into a div whose display style is set to `inline-block`.
2. Set the pointer event as `none` for the disabled element (button) through CSS.
3. Now, initialize the Tooltip for outer div element that holds the disabled button element.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
```

```

        TooltipModule,
        ButtonModule
    ],
    standalone: true,
    selector: 'my-app',
    template: `
        <ejs-tooltip id="tooltip" content='Tooltip from disabled element'>
            <div><input ejs-button type="button" disabled value="Disabled
button" /></div>
        </ejs-tooltip>
    `,
    styles: [ `
        #tooltip {
            display: inline-block;
            position: relative;
            top: 50px;
            left: 40%;
        }
        #tooltip [disabled] {
            pointer-events: none;
            font-size: 22px;
            padding: 10px;
        }
    ` ]
})
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 tooltip content with html elements in Angular Tooltip component

The Tooltip component loads HTML tags using the [content](#) template.

The HTML tags such as `<div>`, ``, **bold**, *italic*, underline, etc., can be used. Style attributes can also be applied with HTML tags.

Here, Bold, Italic, Underline, and Anchor tags are used.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewChild, ViewEncapsulation } from '@angular/core';
@Component({
    imports: [

        TooltipModule, ButtonModule
    ],

```

```

standalone: true,
  selector: 'my-app',
  template: `
    <div id="content">
      <h2>HTML Tags</h2>
      Through templates, <b><span style="color:#e3165b">tooltip
content</span></b> can be loaded with <u><i> inline HTML, images, iframe,
videos, maps </i></u>. A title can be added to the content
    </div>
    <div class="tooltipContent">
      <ejs-tooltip #tooltip id="tooltip" position='BottomCenter'>
        <ng-template #content>
          <div>
            Through templates,<b><span style="color:#e3165b">tooltip
content</span></b> can be loaded with <u><i> inline HTML, images, iframe,
videos, maps </i></u>. A title can be added to the content
          </div>
        </ng-template>
        <input ej-button type="button" class="text" id="Title"
value="HTML(With Title)" />
      </ejs-tooltip>
    </div>
  `,
  encapsulation: ViewEncapsulation.None,
})
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 tooltip with dynamic html in Angular Tooltip component

Tooltip loads HTML pages via HTML tags such as iframe, video, and map using the [content](#) property, which supports both string and HTML tags.

To load an `iframe` element in tooltip, set the required iframe in the `content` of tooltip while initializing the tooltip component. Refer to the following code.

`typescript

```

content= '<iframe src="https://www.syncfusion.com/products/essential-js2"></iframe>'
`

```

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewEncapsulation } from '@angular/core';
import { TooltipComponent } from '@syncfusion/ej2-angular-popups';

```

```

@Component({
  imports: [

    TooltipModule, ButtonModule

  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <div id="tooltipContent">
      <div class="content">
        <ejs-tooltip cssClass='e-tooltip-css' position='BottomCenter'
opensOn='Click'>
          <!-- iframe element -->
          <ng-template #content>
            <iframe
src="https://ej2.syncfusion.com/showcase/typescript/expensetracker/#/dashboa
rd"></iframe>
            </ng-template>
            <button ejs-button class="text" id="iframeContent"
cssClass='e-outline' isPrimary=true>Embedded Iframe</button>
          </ejs-tooltip>
        </div>
      </div>`,
  encapsulation: ViewEncapsulation.None,
})
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));

```

Tooltip open or display modes in Angular Tooltip component

The open mode property of tooltip can be defined on a target that is hovering, focusing, or clicking.

Tooltip component have the following types of open mode:

- Auto
- Hover
- Click
- Focus
- Custom

Auto

Tooltip appears when you hover over the target or when the target element receives the focus.

Hover

Tooltip appears when you hover over the target.

Click

Tooltip appears when you click a target element.

Focus

Tooltip appears when you focus (say through tab key) on a target element.

Custom

Tooltip is not triggered by any default action. So, bind your own events and use either open or close public methods.

APP.COMPONENT.TS

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

    TooltipModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <div id="sample">
      <div id="first">
        <ejs-tooltip id="showTooltip" opensOn='Hover' content='Tooltip from
        hover' position='BottomCenter'>
          <button class="blocks" id="tooltiphover">Hover me</button>
        </ejs-tooltip>
        <ejs-tooltip id="showTooltip" opensOn='Click' content='Tooltip from
        click' position='BottomCenter'>
          <button class="blocks" id="tooltipclick">Click me</button>
        </ejs-tooltip>
      </div>
      <div id="second">
        <ejs-tooltip id="showTooltip" content='Click close icon to close me'
        [isSticky]='true' position='BottomCenter'>
          <button class="blocks" id="tooltipclick">Sticky mode</button>
        </ejs-tooltip>
        <ejs-tooltip id="showTooltip" content='Opens and closes Tooltip with
        delay of <i>1000 milliseconds</i>' position='BottomCenter'
        [openDelay]='1000' [closeDelay]='1000'>
          <button class="blocks" id="tooltipclick">Tooltip with
        delay</button>
        </ejs-tooltip>
      </div>
      <div id="third">
        <ejs-tooltip #tooltip id="showTooltip" content='Tooltip from custom
        mode' opensOn='Custom' position='BottomCenter'
        (dblclick)='customOpen($event)'>
          <button class="blocks" id="tooltipclick">Double Click on
        Me</button>
        </ejs-tooltip>
        <ejs-tooltip #tooltip id="showTooltip" content='Tooltip from focus'
        position='BottomCenter'>
```

```

        <div id="textbox" class="e-float-input">
            <input id="focus" type="text" placeholder="Focus and blur"/>
        </div>
    </ejs-tooltip>
</div>
</div>
`
,
}))
export class AppComponent {
    @ViewChild('tooltip')
    public tooltipControl: TooltipComponent | any ;
    tooltipCustom: any;
    constructor() { }
    customOpen(args: any): void {
        if (args.target.getAttribute("data-tooltip-id")) {
            this.tooltipCustom.close();
        } else {
            this.tooltipCustom.open(args.target);
        }
    }
}
}

```

MAIN.TS

```

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

```

Fancy tooltip customization in Angular Tooltip component

The arrow of the tooltip can be customized as needed by customizing CSS in the sample-side.

The EJ2 tooltip component is achieved through CSS3 format and positioned the tip arrow according to the tooltip positions like **TopCenter**, **BottomLeft**, **RightTop**, and more.

Here, the tip arrow is customized as Curved tooltip and Bubble tooltip.

Curved tip

The content for the tip pointer arrow has been added. To customize the curved tip arrow, override the following CSS class of tip arrow.

```

`typescript
.e-arrow-tip-outer.e-tip-bottom,
.e-arrow-tip-outer.e-tip-top {
border-left: none !important;
border-right: none !important;
border-top: none !important;
}
.e-arrow-tip.e-tip-top {

```



```
transform: rotate(170deg);
}
`
```

Bubble tip

The two `div`s (inner div and outer div) have been added to achieve the bubble tip arrow. To customize the bubble tip arrow, override the following CSS class of tip arrow.

```
`typescript
.e-arrow-tip-outer.e-tip-bottom, .e-arrow-tip-outer.e-tip-top
{
border-radius: 50px;
height: 10px;
width: 10px;
}
.e-arrow-tip-inner.e-tip-bottom, .e-arrow-tip-inner.e-tip-top
{
border-radius: 50px;
height: 10px;
width: 10px;
}
`
```

These tip arrow customizations have been achieved through CSS changes in the sample level. The tooltip position can be changed by using the radio button click event.

The arrow tip pointer can also be disabled by using the [showTipPointer](#) property in a tooltip.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { RadioButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewChild, ViewEncapsulation } from '@angular/core';
import { TooltipComponent } from '@syncfusion/ej2-angular-popups';
import { RadioButtonComponent, ChangeArgs, ButtonComponent } from
 '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [

    TooltipModule, RadioButtonModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
```

```

<div id="customization">
  <ejs-tooltip #tooltipcurve cssClass='curvetips e-tooltip-css'
content='Tooltip arrow customized'>
    <button id="target">
      Curve Tip Arrow
    </button>
  </ejs-tooltip>
</div>
<div id="positions">
  <ul>
    <li><ejs-radiobutton label="TopCenter" value="TopCenter"
name="state" checked='true' (change)="onChange($event)"></ejs-
radiobutton></li>
    <li><ejs-radiobutton label="BottomLeft" value="BottomLeft"
name="state" (change)="onChange($event)"></ejs-radiobutton></li>
  </ul>
</div>
<div id="balloon">
  <ejs-tooltip #tooltip cssClass='bubbletip e-tooltip-css'
content='Tooltip arrow customized as balloon tip' position='TopRight'>
    <button id="bubbletip">
      Bubble Tip Arrow
    </button>
  </ejs-tooltip>
</div>
<div id="btn">
  <ul>
    <li><ejs-radiobutton label="TopRight" value="TopRight"
name="default" [checked]="true" (change)="onChanged($event)"></ejs-
radiobutton></li>
    <li><ejs-radiobutton label="BottomLeft" value="BottomLeft"
name="default" (change)="onChanged($event)"></ejs-radiobutton></li>
  </ul>
</div>
  <ejs-tooltip #tooltip cssClass='pointertip e-tooltip-css'
content='Disabled Tooltip Pointer' mouseTrail='true'
[showTipPointer]='false'>
    <button id="tooltip">
      Disabled Tip Arrow
    </button>
  </ejs-tooltip>
  ,
  encapsulation: ViewEncapsulation.None,
})
export class AppComponent {
  @ViewChild('tooltip')
  public tooltipControl: TooltipComponent | any ;
  @ViewChild('tooltipcurve')
  public tooltipCurve : TooltipComponent | any;
  tooltipCustom: any;
  constructor() { }
  onChange(args: ChangeArgs): void {
    this.tooltipCurve.position = args.value as any;
    this.tooltipCurve.dataBind();
  }
  onChanged(args: ChangeArgs): void {
    this.tooltipCustom.position = args.value as any;

```

```
        if( this.tooltipCustom.position == 'BottomLeft') {
            this.tooltipCustom.offsetY = -30;
        } else {
            this.tooltipCustom.offsetY = 0;
        }
        this.tooltipCustom.dataBind();
    }
}
```

MAIN.TS

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

Display tooltip on svg and canvas elements in Angular Tooltip component

Tooltip can be displayed on both SVG and Canvas elements. You can directly attach the `<svg>` or `<canvas>` elements to show tooltips on data visualization elements.

SVG

Create the SVG square element and refer to the following code snippet to render the tooltip on SVG square.

`typescript

```
<ejs-tooltip cssClass='e-tooltip-css' content='SVG Square' target='#square'>
```

```
<svg>
```

```
<rect id="square" class="shape" x="50" y="20" width="50" height="50"
style="fill:#FDA600;stroke:none;stroke-width:5;stroke-opacity:0.9" />
```

```
</svg>
```

```
</ejs-tooltip>
```

,

Canvas

Create the canvas circle element and refer to the following code snippet to render the tooltip on Canvas circle.

`typescript

```
<ejs-tooltip cssClass='e-tooltip-css' content='Canvas Circle' target='#circle'>
```

```
<canvas #circle id="circle" class="shape" width="60" height="60"></canvas>
```

```
</ejs-tooltip>
```

,

APP.COMPONENT.TS

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

```

import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { Component, ViewChild, ViewEncapsulation, ElementRef } from
 '@angular/core';
import { TooltipComponent } from '@syncfusion/ej2-angular-popups';
@Component({
imports: [

    TooltipModule
],
standalone: true,
selector: 'my-app',
template: `
    <div id="box">
        <div class="circletool" id="rectShape" style="left:1%;top:10%">
            <ejs-tooltip cssClass='e-tooltip-css' content='SVG Square'
target='#square'>
                <svg>
                    <rect id="square" class="shape" x="50" y="20" width="50"
height="50" style="fill:#FDA600;stroke:none;stroke-width:5;stroke-
opacity:0.9" />
                </svg>
            </ejs-tooltip>
        </div>
        <div class="circletool" id="ellipseShape" style="top:63%;">
            <ejs-tooltip cssClass='e-tooltip-css' content='SVG Ellipse'
target='#ellipse'>
                <svg>
                    <ellipse id="ellipse" class="shape" cx="100" cy="50" rx="20"
ry="40" style="fill:#0450C2;stroke:none;stroke-width:2" />
                </svg>
            </ejs-tooltip>
        </div>
        <div class="circletool" id="polyShape" style="top:25%;left:40%">
            <ejs-tooltip cssClass='e-tooltip-css' content='SVG Polyline'
target='#polyline'>
                <svg>
                    <polyline id="polyline" class="shape" points="0,40 40,40
40,80 80,80 80,120 120,120 120,160"
style="fill:#ffffff;stroke:#0450C2;stroke-width:4" />
                </svg>
            </ejs-tooltip>
        </div>
        <div class="circletool" id="circleShape" style="top:16%;left:72%">
            <ejs-tooltip cssClass='e-tooltip-css' content='Canvas Circle'
target='#circle'>
                <canvas #circle id="circle" class="shape" width="60"
height="60"></canvas>
            </ejs-tooltip>
        </div>
        <div class="circletool" id="triShape" style="top:73%;left:76%">
            <ejs-tooltip cssClass='e-tooltip-css' content='Canvas Triangle'
target='#triangle'>
                <canvas #triangle id="triangle" class="shape" width="100"
height="50"></canvas>
            </ejs-tooltip>
        </div>
    </div>
`

```

```

    encapsulation: ViewEncapsulation.None,
  })
  export class AppComponent {
    public context?: any;
    public circlecontext?: any;
    @ViewChild('triangle') canvasRef: ElementRef | any;
    @ViewChild('circle') circleRef: ElementRef | any;
    ngOnInit() {
      if (this.canvasRef.nativeElement.getContext) {
        this.context = this.canvasRef.nativeElement.getContext('2d');
        this.context.beginPath();
        this.context.moveTo(0, 50);
        this.context.lineTo(100, 50);
        this.context.lineTo(50, 0);
        this.context.fillStyle = "#FDA600";
        this.context.fill();
      }
      this.circlecontext = this.circleRef.nativeElement.getContext('2d');
      let centerX: number = this.circleRef.nativeElement.width / 2;
      let centerY: number = this.circleRef.nativeElement.height / 2;
      let radius: number = 30;
      this.circlecontext.beginPath();
      this.circlecontext.arc(centerX, centerY, radius, 0, 2 * Math.PI, false);
      this.circlecontext.fillStyle = '#0450C2';
      this.circlecontext.fill();
    }
  }

```

MAIN.TS

```

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

```

Create and show tooltip on multiple targets in Angular Tooltip component

Tooltip can be created and shown on multiple targets within a container by defining the specific target elements to the target property. So, the tooltip is initialized only on matched targets within a container.

In this case, the tooltip content is assigned from the title attribute of the target element.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewChild } from '@angular/core';
import { TooltipComponent, TooltipModule } from '@syncfusion/ej2-angular-popups';
@Component({
  imports: [

    TooltipModule, ButtonModule

  ],

```

```

standalone: true,
  selector: 'my-app',
  template: `
    <div id="tool">
      <form id="details" role="form">
        <div id="user">
          <div class="info">User Name:</div>
          <ejs-tooltip #tooltip position='RightCenter' title="Please enter
your name">
            <div class="inputs"><input type="text" id="uname" class="e-info e-
input" name="firstname" title="Please enter your name"></div>
          </ejs-tooltip>
        </div>
        <br/>
        <div>
          <div class="info">Email Address:</div>
          <ejs-tooltip position='RightCenter' title="Enter a valid email
address">
            <div class="inputs"><input type="text" id="mail" class="e-info e-
input" name="email" title="Enter a valid email address"></div>
          </ejs-tooltip>
        </div>
        <br/>
        <div>
          <div class="info">Password:</div>
          <ejs-tooltip #tooltippwd position='RightCenter' title="Be at least
8 characters length">
            <div class="inputs"><input id="pwd" type="password" class="e-info
e-input" name="password" title="Be at least 8 characters length"></div>
          </ejs-tooltip>
        </div>
        <br/>
        <div>
          <div class="info">Confirm Password:</div>
          <ejs-tooltip position='RightCenter' title="Re-enter your
password">
            <div class="inputs"><input id="cpwd" type="password" class="e-info
e-input" name="Cpwd" title="Re-enter your password"></div>
          </ejs-tooltip>
        </div>
        <br/>
        <div class="btn">
          <input id="sample" ejs-button type="button" value="Submit"
(click)="submitdata($event)" />
          <input id="clear" ejs-button type="reset" value="Reset"
(click)="cleardata($event)" />
        </div>
      </form>
    </div>
  `
})
export class AppComponent {
  @ViewChild('tooltip')
  tooltip: TooltipComponent | any
  @ViewChild('tooltippwd')
  tooltipcontrol: TooltipComponent | any
  submitdata(event: any) {

```

```

let name = (document.getElementById('uname')! as any).value;
let pwd = (document.getElementById('pwd')! as any).value;
let cpwd = (document.getElementById('cpwd')! as any).value;
if(name.length > 0 && name.length < 4){
    document.getElementById('uname')!.title = 'Required Minimum 4
Characters';
    document.getElementById('uname')!.style.backgroundColor = 'red';
    let target = document.getElementById('uname');
    this.tooltip.open(target);
} else {
    document.getElementById('uname')!.style.backgroundColor = 'white';
}
if(pwd !== '' && pwd.length < 8){
    document.getElementById('pwd')!.title = 'Required Minimum 8
Characters';
    document.getElementById('pwd')!.style.backgroundColor = 'red';
    let pdwtargt = document.getElementById('pwd');
    this.tooltipcontrol.open(pdwtargt);
} else {
    document.getElementById('pwd')!.style.backgroundColor = 'white';
}
if(name.length >= 4 && pwd !== '' && pwd.length >= 8 && pwd == cpwd ){
    alert('Form Submitted');
} else {
    alert('Details are not Valid');
}
}
}
cleardata(event: any){
    document.getElementById('uname')!.style.backgroundColor = 'white';
    document.getElementById('pwd')!.style.backgroundColor = 'white';
    let target = document.getElementById('uname');
    this.tooltip.close(target);
    document.getElementById('uname')!.title = 'Please enter your name';
    let pdwtargt = document.getElementById('pwd');
    this.tooltipcontrol.close(pdwtargt);
}
}

```

MAIN.TS

```

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

```

Dynamic tooltip content in Angular Tooltip component

The tooltip content can be changed dynamically using the Fetch request.

The Fetch request should be made within the [beforeRender](#) event of the tooltip. On every success, the corresponding retrieved data will be set to the [content](#) property of the tooltip.

When you hover over the icons, its respective data will be retrieved dynamically and then assigned to the tooltip's content.

Refer to the following code snippet to change the tooltip content dynamically.

```

`typescript
onBeforeRender(args: TooltipEventArgs): void {
  this.content = 'Loading...';
  this.dataBind();
  let fetchApi: Fetch = new Fetch('./tooltip.json', 'GET');
  fetchApi.send().then(
    (result: any) => {
      for (let i: number = 0; i < result.length; i++) {
        if (result[i].Id == args.target.id) {
          / tslint:disable /
          this.content = result[i].Name;
          / tslint:enable /
        }
      }
      this.dataBind();
    },
    (reason: any) => {
      this.content = reason.message;
      this.dataBind();
    }
  );
}
`

```

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { RadioButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewChild } from '@angular/core';
import { Fetch } from '@syncfusion/ej2-base';
import { TooltipComponent, TooltipEventArgs, TooltipModule } from
 '@syncfusion/ej2-angular-popups';
@Component({
  imports: [
    TooltipModule, RadioButtonModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <div id="tool">

```



```

    <h2> Dynamic Tooltip content </h2>
    <ejs-tooltip #tooltip id='tooltip' content='Loading...'
target=".circletool" [showTipPointer]='false'
(beforeRender)="onBeforeRender($event)">
        <div id="box">
            <div id='1' class="circletool bold-01"></div>
            <div id='2' class="circletool italic"></div>
            <div id='3' class="circletool underline-02"></div>
            <div id='4' class="circletool cut-02"></div>
            <div id='5' class="circletool copy"></div>
            <div id='6' class="circletool paste"></div>
        </div>
    </ejs-tooltip>
</div>
` ,
}))
export class AppComponent {
    @ViewChild('tooltip')
    public tooltipControl: TooltipComponent | any;
    constructor() {}
    onBeforeRender(args: TooltipEventArgs): void {
        this.tooltipControl.content = 'Loading...';
        this.tooltipControl.dataBind();
        let fetchApi: Fetch = new Fetch('./tooltipdata.json', 'GET');
        fetchApi.send().then(
            (result: any) => {
                for (let i: number = 0; i < result.length; i++) {
                    if (result[i].Id == args.target.id) {
                        /* tslint:disable */
                        this.tooltipControl.content = result[i].Name;
                        /* tslint:enable */
                    }
                }
                this.tooltipControl.dataBind();
            },
            (reason: any) => {
                this.tooltipControl.content = reason.message;
                this.tooltipControl.dataBind();
            }
        );
    }
}

```

MAIN.TS

```

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

```

Tooltip content template in Angular Tooltip component

The Tooltip component [content](#) can be loaded through template support. Refer to the below code snippet.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TooltipModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewChild, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [
    TooltipModule, ButtonModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `
    <ejs-tooltip style="display:block;position:absolute;left:calc( 50% - 60px);top:45%;">
      <button ejs-button >Show Tooltip</button>
      <ng-template #content >
        Tooltip content here!!!
      </ng-template>
    </ejs-tooltip>
  `,
  encapsulation: ViewEncapsulation.None
})
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));
```

TreeGrid

Getting started with Angular Treegrid component

This section explains the steps required to create a simple Essential JS 2 TreeGrid and demonstrates the basic usage of the [Link to the Video](#) in a Angular CLI application.

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

Setup Angular Environment

You can use the [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 the Angular CLI command as follows.

```
`bash
```

```
ng new my-app
```

```
cd my-app
```

```
,
```

Installing Syncfusion TreeGrid 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-treegrid](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-treegrid --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-treegrid@ngcc](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-treegrid@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-treegrid:"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 TreeGrid Module

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

```
`typescript
```

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

```
import { BrowserModule } from '@angular/platform-browser';
// import the TreeGridModule for the TreeGrid component
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid';
import { AppComponent } from './app.component';
@NgModule({
  //declaration of ej2-angular-treegrid module into NgModule
  imports: [ BrowserModule, TreeGridModule ],
  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 the 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-calendars/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-navigations/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-grids/styles/material.css';
@import '../node_modules/@syncfusion/ej2-treegrid/styles/material.css';
```

Add TreeGrid component

Modify the template in `[src/app/app.component.ts]` file to render the TreeGrid component.

Add the Angular TreeGrid by using `<ejs-treegrid>` selector in `template` section of the `app.component.ts` file.

```
`typescript
import { Component, OnInit } from '@angular/core';
@Component({
```

```
selector: 'app-container',  
// specifies the template string for the TreeGrid component  
template: <ejs-treegrid> </ejs-treegrid>  
})  
export class AppComponent implements OnInit {  
  ngOnInit(): void {  
  }  
}
```

Defining Row Data

Bind data for the TreeGrid component by using `dataSource` property.

It accepts either an array of JavaScript object or `DataManager` instance.

```
`typescript  
import { Component, OnInit } from '@angular/core';  
import { sampleData } from './datasource';  
@Component({  
  selector: 'app-container',  
  template: <ejs-treegrid [dataSource]='data'> </ejs-treegrid>  
})  
export class AppComponent implements OnInit {  
  public data: Object[];  
  ngOnInit(): void {  
    this.data = sampleData;  
  }  
}
```

Defining Columns

The TreeGrid has an option to define the columns as an array. In these columns, the following properties are used to customize the columns.

- The `field` has been added to map with a property name in an array of JavaScript objects.
- The `headerText` has been added to change the title of columns.
- The `textAlign` has been used to change the alignment of columns. By default, columns will be left aligned. To change the columns to right align, define the `textAlign` to `Right`.

- Also, the another useful property, `format` has been used. Using this, you can format number and date values to standard or custom formats. Here it is defined for the conversion of date objects to formatted strings.

Tree Column is used to expand or collapse child rows is defined by using the [treeColumnIndex](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
@Component({
  imports: [
    TreeGridModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
childMapping='subtasks'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=70></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=200></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  ngOnInit(): void {
    this.data = sampleData;
  }
}
```

MAIN.TS

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

In the above code example, the hierarchical data binding is represented in which the [childMapping](#) property denotes the hierarchy relationship; whereas in self-referencing data binding [idMapping](#) and [parentIdMapping](#) denotes the hierarchy relationship.

Module injection

To create TreeGrid with additional features, inject the required modules. The following modules are used to extend TreeGrid's basic functionality.

- **PageService** - Inject this module to use paging feature.
- **SortService** - Inject this module to use sorting feature.
- **FilterService** - Inject this module to use filtering feature.
- **ExcelExportService** - Inject this module to use Excel export feature.
- **PdfExportService** - Inject this module to use PDF export feature.

These modules should be injected into the **providers** section of root **NgModule** or component class.

Additional feature modules are available [here](#)

Enable Paging

The paging feature enables users to view the TreeGrid record in a paged view. It can be enabled by setting the [allowPaging](#) property to true. Also, need to inject the **PageService** module in the **providers** section as follows. If the **PageService** module is not injected, the pager will not be rendered in the TreeGrid. The pager can be customized using the [pageSettings](#) property.

In root-level paging mode, paging is based on the root-level rows only i.e., it ignores the child rows count and it can be enabled by using the [pageSettings.pageSizeMode](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
import { PageSettingsModel } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [

    TreeGridModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
childMapping='subtasks'
[allowPaging]="true" [pageSettings]='pageSettings'>
  <e-columns>
    <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
```

```

        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public pageSettings?: PageSettingsModel;
        ngOnInit(): void {
            this.data = sampleData;
            this.pageSettings = { pageSize: 6 };
        }
    }
}

```

MAIN.TS

```

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

```

Enable Sorting

The sorting feature enables the user to order the records.

It can be enabled by setting [allowSorting](#) property to true.

Also, need to inject the [SortService](#) module in the provider section as follow.

If we didn't inject the [SortService](#) module, then user not able to sort when click on headers.

Sorting feature can be customized using [sortSettings](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
import { PageSettingsModel, SortSettingsModel } from '@syncfusion/ej2-
angular-treegrid';
@Component({
    imports: [

        TreeGridModule
    ],
    providers: [PageService,
        SortService,
        FilterService],
    standalone: true,

```



```

        selector: 'app-container',
        template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
[sortSettings]="sortSettings"
        [allowSorting]="true" childMapping='subtasks'
[allowPaging]="true" [pageSettings]='pageSettings'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
                <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
                <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
            </e-columns>
        </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public sortSettings?: SortSettingsModel;
        public pageSettings?: PageSettingsModel;
        ngOnInit(): void {
            this.data = sampleData;
            this.sortSettings = { columns: [{ field: 'taskName', direction:
'Ascending' }, { field: 'taskID', direction: 'Descending' }] };
            this.pageSettings = { pageSize: 6 };
        }
    }

```

MAIN.TS

```

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

```

Enable Filtering

The filtering feature enables you to view the reduced amount of records based on the filter criteria. It can be enabled by setting the [allowFiltering](#) property to true. Also, need to inject the `FilterService` module in the provider section as follow. If `FilterService` module is not injected, filter bar will not be rendered in TreeGrid. Filtering feature can be customized using the [filterSettings](#) property.

By default, filtered records are shown along with its parent records. This behavior can be changed by using the [filterSettings-hierarchyMode](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';

```

```

import {PageSettingsModel, SortSettingsModel } from '@syncfusion/ej2-
angular-treegrid';
@Component({
imports: [

    TreeGridModule
],
providers: [PageService,
            SortService,
            FilterService],
standalone: true,
selector: 'app-container',
template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
[allowFiltering]="true" [sortSettings]="sortSettings"
[allowSorting]="true" childMapping='subtasks'
[allowPaging]="true" [pageSettings]='pageSettings'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`
})
export class AppComponent implements OnInit {
    public data?: Object[];
    public sortSettings?: SortSettingsModel;
    public pageSettings?: PageSettingsModel;
    ngOnInit(): void {
        this.data = sampleData;
        this.sortSettings = { columns: [{ field: 'taskName', direction:
'Ascending' }, { field: 'taskID', direction: 'Descending' }] };
        this.pageSettings = { pageSize: 6 };
    }
}

```

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 render the records in a collapsed state in the initial render of the tree grid by enabling the [enableCollapseAll](#) property.

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 { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
import { PageSettingsModel, SortSettingsModel } from '@syncfusion/ej2-
angular-treegrid';
@Component({
  imports: [

    TreeGridModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
[sortSettings]="sortSettings"
[allowFiltering]="true" [allowSorting]="true"
childMapping='subtasks' [allowPaging]="true"
[pageSettings]='pageSettings'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public sortSettings?: SortSettingsModel;
  public pageSettings?: PageSettingsModel;
  ngOnInit(): void {
    this.data = sampleData;
    this.sortSettings = { columns: [{ field: 'taskName', direction:
'Ascending' }, { field: 'taskID', direction: 'Descending' }] };
    this.pageSettings = { pageSize: 6 };
  }
}
```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Modules in Angular Treegrid component

The following value providers should be injected to extend TreeGrid's functionality.

Module	Description
----- -----	
PageService	Inject this module to use paging feature.
SortService	Inject this module to use sorting feature.
FilterService	Inject this module to use filtering feature.
EditService	Inject this module to use editing feature.
AggregateService	Inject this module to use aggregate feature.
ColumnMenuService	Inject this module to use column menu feature.
CommandColumnService	Inject this module to use command column feature.
ContextMenuService	Inject this module to use context menu feature.
ResizeService	Inject this module to use resize feature.
ReorderService	Inject this module to use reorder feature.
PrintService	Inject this module to use to use print feature and this is a default injected module.
ToolbarService	Inject this module to use toolbar feature.
ExcelExportService	Inject this module to use Excel export feature.
PdfExportService	Inject this module to use PDF export feature.
RowDDService	Inject this module to use Indent feature.

These modules should be injected into the `providers` section of root `NgModule` or component class.

You can get the data module by using the [getDataModule](#) method in tree grid.

You can refer to our [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Data Binding

Data binding in Angular Treegrid component

The TreeGrid uses `DataManager`, which supports both RESTful JSON data services binding and local JavaScript object array binding. The [dataSource](#) property can be assigned either with the instance of [Link to the Video](#) or JavaScript object array collection.

It supports two kinds of data binding method:

- Local data
- Remote data

To learn about how to bind local or remote data to Tree Grid, you can check on this video:

[Binding with ajax](#)

You can use TreeGrid [dataSource](#) property to bind the data source to TreeGrid from external fetch request. In the below code we have fetched the data source from the server with the help of fetch request and provided that to [dataSource](#) property by using `onSuccess` event of the fetch.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit, ViewChild } from '@angular/core';
import { Fetch } from '@syncfusion/ej2-base';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { DataManager } from '@syncfusion/ej2-data';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: ` <button ej2-button (click)="click()">Bind Data</button>
    <ejs-treegrid #treegrid [dataSource]='data' [treeColumnIndex]='1'
    parentIdMapping='ParentItem' idMapping='TaskID' [allowPaging]="true"
    height=240>
      <e-columns>
        <e-column field='TaskID' headerText='Task ID' width='90'
        textAlign='Right'></e-column>
        <e-column field='TaskName' headerText='Task Name'
        width='170'></e-column>
        <e-column field='StartDate' headerText='Start Date' width='130'
        format="yMd" textAlign='Right'></e-column>
        <e-column field='Duration' headerText='Duration' width='80'
        textAlign='Right'></e-column>
      </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: DataManager;
  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  ngOnInit(): void {
  }
  click(): any{
```

```

    let fetch = new
Fetch("https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData", "GET");
    let trgrid = this.treegrid;
    fetch.send();
    fetch.onSuccess = function (data: string) {
        (trgrid as TreeGridComponent).hideSpinner();
        (trgrid as TreeGridComponent).dataSource = data;
    };
}
}

```

MAIN.TS

```

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

```

* If you bind the dataSource from this way, then it acts like a local dataSource. So you cannot perform any server side crud actions.

Handling expandStateMapping

To denotes the expand status of parent row, define the [expandStateMapping](#) property of tree grid.

The `expandStateMapping` property maps the field name in data source, that denotes whether parent record is in expanded or collapsed state and this is useful to renders parent row in expanded or collapsed state based on this mapping property value in data source.

`typescript

```

import { Component, OnInit } from '@angular/core';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
import './App.css';

@Component({
    selector: 'app-container',
    template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1' height='400'
    hasChildMapping='isParent', idMapping='TaskID' expandStateMapping='IsExpanded'
    parentIdMapping='ParentValue'>
    <e-columns>
    <e-column field='TaskID' headerText='Task ID' width='90' textAlign='Right'></e-column>
    <e-column field='TaskName' headerText='Task Name' width='180'></e-column>
    <e-column field='Duration' headerText='Duration' width='80' textAlign='Right'></e-column>
    </e-columns>
    </ejs-treegrid>`
})

```

```

export class AppComponent implements OnInit {
  public data: DataManager;
  public dataManager: DataManager = new DataManager({
    adaptor: new UrlAdaptor,
    url: "Home/DataSource",
  });
  ngOnInit(): void {
    this.data = this.dataManager;
  }
}

```

The following code example defines `expandStateMapping` property at server end.

```

`typescript
public ActionResult ExpandStateMapping()
{
  return View();
}

public class TreeData
{
  public static List<TreeData> tree = new List<TreeData>();
  [System.ComponentModel.DataAnnotations.Key]
  public int TaskID { get; set; }
  public string TaskName { get; set; }
  public int Duration { get; set; }
  public int? ParentValue { get; set; }
  public bool? isParent { get; set; }
  public bool IsExpanded { get; set; }
  public TreeData() { }
  public static List<TreeData> GetTree()
  {
    if (tree.Count == 0)
    {
      int root = 0;

```

```
for (var t = 1; t <= 500; t++)
{
    Random ran = new Random();
    string math = (ran.Next() % 3) == 0 ? "High" : (ran.Next() % 2) == 0 ? "Release Breaker" : "Critical";
    string progr = (ran.Next() % 3) == 0 ? "Started" : (ran.Next() % 2) == 0 ? "Open" : "In Progress";
    root++;
    int rootItem = root;
    tree.Add(new TreeData() { TaskID = rootItem, TaskName = "Parent task " + rootItem.ToString(), isParent
    = true, IsExpanded = false, ParentValue = null, Duration = ran.Next(1, 50) });
    int parent = root;
    for (var d = 0; d < 1; d++)
    {
        root++;
        string value = ((parent + 1) % 3 == 0) ? "Low" : "Critical";
        int par = parent + 1;
        progr = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";
        int iD = root;
        tree.Add(new TreeData() { TaskID = iD, TaskName = "Child task " + iD.ToString(), isParent = true,
        IsExpanded = false, ParentValue = rootItem, Duration = ran.Next(1, 50) });
        int subparent = root;
        for (var c = 0; c < 500; c++)
        {
            root++;
            string val = ((subparent + c + 1) % 3 == 0) ? "Low" : "Critical";
            int subchild = subparent + c + 1;
            string progress = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";
            int childID = root ;
            tree.Add(new TreeData() { TaskID = childID, TaskName = "sub Child task " + childID.ToString(), isParent =
            false, IsExpanded = false, ParentValue = subparent, Duration = ran.Next(1, 50) });
        }
    }
}
return tree;
```



```

}
}
`

```

You can refer to our [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Local data in Angular Treegrid component

In Local Data binding, data source for rendering the TreeGrid control is retrieved from the same application locally.

Two types of Data binding are possible with the TreeGrid control.

- Hierarchical Datasource binding
- Self-Referential Data binding (Flat Data)

To bind local data to the treegrid, you can assign a JavaScript object array to the [dataSource](#) property. The local data source can also be provided as an instance of the **DataManager**.

By default, **DataManager** uses **JsonAdaptor** for local data-binding.

Hierarchy data source binding

The [childMapping](#) property is used to map the child records in hierarchy data source.

The following code example shows you how to bind the hierarchical local data into the TreeGrid control.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
import { PageSettingsModel } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
  childMapping='subtasks'
    [allowPaging]="true" [pageSettings]='pageSettings'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
  textAlign='Right' width=90></e-column>

```

```

        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public pageSettings?: PageSettingsModel;
        ngOnInit(): void {
            this.data = sampleData;
            this.pageSettings = { pageSize: 6 };
        }
    }
}

```

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 is not supported for Hierarchy Data.

Self-Referential data binding (Flat data)

TreeGrid is rendered from Self-Referential data structures by providing two fields, ID field and parent ID field.

- **ID Field:** This field contains unique values used to identify nodes. Its name is assigned to the [idMapping](#) property.
- **Parent ID Field:** This field contains values that indicate parent nodes. Its name is assigned to the [parentIdMapping](#) property.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit } from '@angular/core';
import { projectData } from './datasource';
@Component({
    imports: [

        TreeGridModule,
        ButtonModule
    ],
    providers: [PageService,
        SortService,

```

```

        FilterService],
standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
parentIdMapping='parentID' idMapping='TaskID' height=265
[allowPaging]="true">
    <e-columns>
        <e-column field='TaskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
        <e-column field='TaskName' headerText='Task Name'
width='170'></e-column>
        <e-column field='StartDate' headerText='Start Date' width='130'
format="yMd" textAlign='Right'></e-column>
        <e-column field='Duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
    </e-columns>
    </ejs-treegrid>`
  })
  export class AppComponent implements OnInit {
    public data?: Object[];
    ngOnInit(): void {
      this.data = projectData;
    }
  }
}

```

MAIN.TS

```

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

```

Herewith we have provided list of reserved properties and the purpose used in TreeGrid. We recommend to avoid these reserved properties for Internal purpose(To get rid of conflicts).

Reserved keywords | Purpose

childRecords | Specifies the childRecords of a parentData

hasChildRecords | Specifies whether the record contains child records

hasFilteredChildRecords | Specifies whether the record contains filtered child records

expanded | Specifies whether the child records are expanded

parentItem | Specifies the parentItem of childRecords

index | Specifies the index of current record

level | Specifies the hierarchy level of record

filterLevel | Specifies the hierarchy level of filtered record

parentIdMapping | Specifies the parentID

uniqueID | Specifies the unique ID of a record

parentUniqueID | Specifies the parent Unique ID of a record

checkboxState | Specifies the checkbox state of a record

isSummaryRow | Specifies the summary of a record

taskData | Specifies the main data

primaryParent | Specifies the Primary data

Refresh the data source

Add or delete the data source through an external button. To reflect the data source changes in the tree grid, you need to invoke the [refresh](#) method.

Please follow the below steps to refresh the tree grid after changing the data source:

Step 1:

Add or delete the datasource record by using the following code:

```
`typescript
this.treegrid.dataSource.unshift(data); // Add a new record.
this.treegrid.dataSource.splice(selectedRow, 1); // Delete a record.
`
```

Step 2:

Refresh the tree grid after changing the datasource by using the `refresh` method.

```
`typescript
this.treegrid.refresh(); // Refresh the tree grid.
`
```

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { sampleData } from './datasource';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<button ej-button class="e-flat" (click)="add()">Add</button>`
})
```

```

        <button ej-button class="e-flat"
(click)="delete()">Delete</button>
        <ejs-treegrid #treegrid [dataSource]="data"
childMapping="subtasks" height="350" [treeColumnIndex]="1"
[allowPaging]="true" [pageSettings]="pageSettings">
            <e-columns>
                <e-column field="taskID" headerText="Task ID"
width="90" textAlign="Right"></e-column>
                <e-column field="taskName" headerText="Task Name"
width="200"></e-column>
                <e-column field="startDate" headerText="Start Date"
width="110" format="yMd" textAlign="Right"></e-column>
                <e-column field="endDate" headerText="End Date"
width="110" format="yMd" textAlign="Right"></e-column>
                <e-column field="duration" headerText="Duration"
width="100" textAlign="Right"></e-column>
                <e-column field="progress" headerText="Progress"
width="80" textAlign="Right"></e-column>
                <e-column field="priority" headerText="Priority"
width="90"></e-column>
            </e-columns>
        </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data: Object[] = [];
        @ViewChild('treegrid') public treegrid?: TreeGridComponent;
        pageSettings: any;
        ngOnInit(): void {
            this.data = sampleData;
        }
        add(): void {
            const rdata: object = {
                taskID: 102,
                taskName: 'New record',
                startDate: new Date(8367642e5),
                endDate: new Date(8467642e5),
                duration: 15,
                progress: 100,
                priority: 'Normal',
            };
            (this.treegrid?.dataSource as object[]).unshift(rdata);
            this.treegrid?.refresh();
        }
        delete(): void {
            const selectedRow: number = this.treegrid?.getSelectedRowIndex() [0] as
number;
            if (this.treegrid?.getSelectedRowIndex().length) {
                (this.treegrid?.dataSource as object[]).splice(selectedRow, 1);
            } else {
                alert('No records selected for delete operation');
            }
            this.treegrid?.refresh();
        }
    }
}

```

MAIN.TS

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

Get the content of the tree grid by using the [getContent](#) method.

Get the table content by using the [getContentTable](#) method in the tree grid.

Destroy the component by using the [destroy](#) method in the tree grid.

Remote data in Angular Treegrid component

To bind remote data to TreeGrid component, assign service data as an instance of **DataManager** to the [dataSource](#) property. To interact with remote data source, provide the endpoint **url** and define the [hasChildMapping](#) property of treegrid.

The [hasChildMapping](#) property maps the field name in data source, that denotes whether current record holds any child records. This is useful internally to show expand icon while binding child data on demand.

The TreeGrid provides **Load on Demand** support for rendering remote data. The Load on demand is considered in TreeGrid for the following actions.

- Expanding root nodes.
- Navigating pages, with paging enabled in TreeGrid.

When load on demand is enabled, all the root nodes are rendered in collapsed state at initial load.

When load on demand support is enabled in TreeGrid with paging, the current or active page's root node alone will be rendered in collapsed state. On expanding the root node, the child nodes will be loaded from the remote server.

When a root node is expanded, its child nodes are rendered and are cached locally, such that on consecutive expand/collapse actions on root node, the child nodes are loaded from the cache instead from the remote server.

Similarly, if the user navigates to a new page, the root nodes of that specific page, will be rendered with request to the remote server.

Remote Data Binding supports only Self-Referential Data and by default the **pageSizeMode** for Remote Data is **Root** mode. i.e only root node's count will be shown in pager while using Remote Data

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit } from '@angular/core';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
@Component({
  imports: [
```

```

        TreeGridModule,
        ButtonModule
    ],
    providers: [PageService,
                SortService,
                FilterService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
hasChildMapping='isParent' parentIdMapping='ParentItem' idMapping='TaskID'
[allowPaging]="true">
    <e-columns>
        <e-column field='TaskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
        <e-column field='TaskName' headerText='Task Name '
width='170'></e-column>
        <e-column field='StartDate' headerText='Start Date' width='130'
format="yMd" textAlign='Right'></e-column>
        <e-column field='Duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
    </e-columns>
    </ejs-treegrid>`
  })
  export class AppComponent implements OnInit {
    public data?: DataManager;
    ngOnInit(): void {
      this.data = new DataManager({
        url: 'https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData',
        adaptor: new WebApiAdaptor, crossDomain: true
      });
    }
  }
}

```

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, **DataManager** uses **ODataAdaptor** for remote data-binding.

Based on the RESTful web services, set the corresponding adaptor to **DataManager**. Refer [here](#) for more details.

Filtering and searching server-side data operations are not supported in load on demand

LoadChildOnDemand

While binding remote data to Tree Grid component, by default Tree Grid renders parent rows in collapsed state. Tree Grid provides option to load the child records also during the initial rendering itself for remote data binding by setting [loadChildOnDemand](#) as true.

When [loadChildOnDemand](#) is enabled parent records are rendered in expanded state.

The following code example describes the behavior of the loadChildOnDemand feature of Tree Grid.

```
`typescript
import { Component, OnInit } from '@angular/core';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
@Component({
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1' height='270' idMapping='TaskID'
    parentIdMapping='parentID' loadChildOnDemand='true' allowPaging='true'>
    <e-columns>
    <e-column field='TaskID' headerText='Task ID' width='90' textAlign='Right'></e-column>
    <e-column field='TaskName' headerText='Task Name' width='170'></e-column>
    <e-column field='StartDate' headerText='Start Date' width='130' format="yMd" textAlign='Right'></e-
    column>
    <e-column field='EndDate' headerText='End Date' width='130' format="yMd" textAlign='Right'></e-
    column>
    <e-column field='Progress' headerText='Progress' width='100' textAlign='Right'></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data: DataManager;
  public dataManager: DataManager = new DataManager({
    url: "Home/DataSource",
    updateUrl: "Home/Update",
    insertUrl: "Home/Insert",
    removeUrl: "Home/Delete",
    batchUrl: "Home/Remove",
    adaptor: new UrlAdaptor
  });
  ngOnInit(): void {
    this.data = this.dataManager;
  }
}
```

Also while using **loadChildOnDemand** we need to handle the child records on server end and it is applicable to CRUD operations also.

The following code example describes handling of child records at server end.

```
`typescript
public ActionResult UrlDatasource(DataManagerRequest dm)
{
    if (TreeData.tree.Count == 0)
        TreeData.GetTree();
    IEnumerable DataSource = TreeData.tree;
    DataOperations operation = new DataOperations();
    if (dm.Where != null && dm.Where.Count > 0)
    {
        DataSource = operation.PerformFiltering(DataSource, dm.Where, "and"); //perform filtering
    }
    if (dm.Sorted != null && dm.Sorted.Count > 0)
    {
        DataSource = operation.PerformSorting(DataSource, dm.Sorted); //perform sorting
    }
    var count = DataSource.ToList<TreeData>().Count();
    if (dm.Skip != 0)
    {
        DataSource = operation.PerformSkip(DataSource, dm.Skip); //Paging
    }
    if (dm.Take != 0)
    {
        DataSource = operation.PerformTake(DataSource, dm.Take);
    }
    if (dm.Where != null)
    {
        DataSource = CollectChildRecords(DataSource, dm); //method to collect child records
    }
    return dm.RequiresCounts ? Json(new { result = DataSource, count = count }) : Json(DataSource);
}
```

```

public IEnumerable CollectChildRecords(IEnumerable datasource, DataManagerRequest dm)
{
    DataOperations operation = new DataOperations();
    IEnumerable DataSource = TreeData.tree; //use the total DataSource here
    string IdMapping = "TaskID"; //define your IdMapping field name here
    int[] TaskIds = new int[0];
    foreach (var rec in datasource)
    {
        int taskid = (int)rec.GetType().GetProperty(IdMapping).GetValue(rec);
        TaskIds = TaskIds.Concat(new int[] { taskid }).ToArray(); //get the Parentrecord Ids based on
        IdMapping Field
    }
    IEnumerable ChildRecords = null;
    foreach (int id in TaskIds)
    {
        dm.Where[0].value = id;
        IEnumerable records = operation.PerformFiltering(DataSource, dm.Where, dm.Where[0].Operator);
        //perform filtering to collect the childrecords based on Ids
        ChildRecords = ChildRecords == null || (ChildRecords.AsQueryable().Count() == 0) ? records :
        ((IEnumerable<object>)ChildRecords).Concat((IEnumerable<object>)records); //concat the
        childrecords with dataSource
    }
    if (ChildRecords != null)
    {
        ChildRecords = CollectChildRecords(ChildRecords, dm); // repeat the operation for inner level child
        if (dm.Sorted != null && dm.Sorted.Count > 0) // perform Sorting
        {
            ChildRecords = operation.PerformSorting(ChildRecords, dm.Sorted);
        }
        datasource = ((IEnumerable<object>)datasource).Concat((IEnumerable<object>)ChildRecords);
        //concat the childrecords with dataSource
    }
    return datasource;
}

```

Offline mode

On remote data binding, all treegrid actions such as paging, loading child on-demand, will be processed on server-side. To avoid postback, set the treegrid to load all data on initialization and make the actions process in client-side. To enable this behavior, use the `offline` property of `DataManager`.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit } from '@angular/core';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
  parentIdMapping='ParentItem' idMapping='TaskID' [allowPaging]="true">
    <e-columns>
      <e-column field='TaskID' headerText='Task ID' width='90'
  textAlign='Right'></e-column>
      <e-column field='TaskName' headerText='Task Name'
  width='170'></e-column>
      <e-column field='StartDate' headerText='Start Date' width='130'
  format="yMd" textAlign='Right'></e-column>
      <e-column field='Duration' headerText='Duration' width='80'
  textAlign='Right'></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: DataManager;
  ngOnInit(): void {
    this.data = new DataManager({
      url: 'https://ej2services.syncfusion.com/production/web-
  services/api/SelfReferenceData',
      adaptor: new WebApiAdaptor, crossDomain: true, offline: true
    });
  }
}
```

MAIN.TS

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

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

Custom adaptor

You can create your own adaptor by extending the built-in adaptors. The following demonstrates custom adaptor approach and how to add a serial number for the records by overriding the built-in response processing using the `processResponse` method of the `ODataAdaptor`.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit } from '@angular/core';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
parentIdMapping='ParentItem' idMapping='TaskID' height=265
[allowPaging]="true">
    <e-columns>
      <e-column field='Sno' headerText='SNO' width='60'
textAlign='Right'></e-column>
      <e-column field='TaskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
      <e-column field='TaskName' headerText='Task Name'
width='170'></e-column>
      <e-column field='StartDate' headerText='Start Date' width='130'
format="yMd" textAlign='Right'></e-column>
      <e-column field='Duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: DataManager;
  ngOnInit(): void {
    class SerialNoAdaptor extends WebApiAdaptor {
      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;
    }
}

this.data = new DataManager({
    url: 'https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData',
    adaptor: new SerialNoAdaptor, crossDomain: true, offline:
true
});
}
}

```

MAIN.TS

```

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

```

Sending additional parameters to the server

To add a custom parameter to the data request, use the `addParams` method of `Query` class. Assign the `Query` object with additional parameters to the treegrid `query` property.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query, WebApiAdaptor } from '@syncfusion/ej2-data';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
[query]='query' parentIdMapping='ParentItem' idMapping='TaskID' height=265
[allowPaging]="true">
    <e-columns>
      <e-column field='TaskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
      <e-column field='TaskName' headerText='Task Name'
width='170'></e-column>
      <e-column field='StartDate' headerText='Start Date' width='130'
format='yMd' textAlign='Right'></e-column>
    </e-columns>
  `
})

```

```

        <e-column field='Duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
    </e-columns>
</ejs-treegrid>`
}))
export class AppComponent implements OnInit {
    public data?: DataManager;
    public query?: Query;
    ngOnInit(): void {
        this.data = new DataManager({
            url: 'https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData',
            adaptor: new WebApiAdaptor, crossDomain: true
        });
        this.query = new Query().addParams('ej2treegrid', 'true');
    }
}

```

MAIN.TS

```

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

```

Handling HTTP error

During server interaction from the treegrid, some server-side exceptions may occur, and you can acquire those error messages or exception details in client-side using the [actionFailure](#) event.

The argument passed to the [actionFailure](#) event contains the error details returned from the server.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit, ViewChild } from '@angular/core';
import { DataManager } from '@syncfusion/ej2-data';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
    imports: [

        TreeGridModule,
        ButtonModule
    ],
    providers: [PageService,
        SortService,
        FilterService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid #treegrid [treeColumnIndex]='1'
(actionFailure)="onActionFailure($event)"`

```

```

        parentIdMapping='ParentItem' [dataSource]='data' idMapping='TaskID'
[allowPaging]="true">
        <e-columns>
            <e-column field='TaskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
            <e-column field='TaskName' headerText='Task Name'
width='170'></e-column>
            <e-column field='StartDate' headerText='Start Date' width='130'
format="yMd" textAlign='Right'></e-column>
            <e-column field='Duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
        </e-columns>
    </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: DataManager;
        @ViewChild('treegrid')
        public treegrid?: TreeGridComponent;
        ngOnInit(): void {
            this.data = new DataManager({
                url: 'http://some.com/invalidUrl'
            });
        }
        onActionFailure(e: any): any {
            let span: HTMLElement = document.createElement('span');
            ((this.treegrid?.element as HTMLElement).parentNode as
            ParentNode).insertBefore(span, (this.treegrid as
            TreeGridComponent).element);
            span.style.color = '#FF0000';
            span.innerHTML = 'Server exception: 404 Not found';
        }
    }
}

```

MAIN.TS

```

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

```

The [actionFailure](#) event will be triggered not only for the server errors, but also when there is an exception while processing the treegrid actions.

Load on demand with virtualization

While binding remote data to Tree Grid component, by default Tree Grid renders parent rows in collapsed state. When expanding the root node, the child nodes will be loaded from the remote server.

When using virtualization with remote data binding, it helps you to improve the tree grid performance while loading a large set of data by setting [enableVirtualization](#) as true. The Tree Grid UI virtualization allows it to render only rows and columns visible within the view-port without buffering the entire datasource.

[hasChildMapping](#) property maps the field name in data source, that denotes whether current record holds any child records. This is useful internally to show expand icon while binding child data on demand.

`typescript

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

import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';

import { EditSettingsModel, ToolbarItems, VirtualScrollService, ToolbarService, PageService,
FilterService, EditService, SortService } from '@syncfusion/ej2-angular-treegrid';

@Component({
  selector: 'app-container',

  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1' height='400' idMapping='TaskID'
  parentIdMapping='ParentValue' hasChildMapping='isParent' expandStateMapping='IsExpanded'
  loadChildOnDemand='true' enableVirtualization='true' allowFiltering='true' allowSorting='true'
  [allowPaging]='true' [pageSettings]='pageSettings' [toolbar]='toolbarOptions'
  [editSettings]='editSettings'>
    <e-columns>
    <e-column field='TaskID' headerText='Task ID' width='90' textAlign='Right'></e-column>
    <e-column field='TaskName' headerText='Task Name' width='180'></e-column>
    <e-column field='Duration' headerText='Duration' width='80' textAlign='Right'></e-column>
    </e-columns>
  </ejs-treegrid>`,

  providers: [VirtualScrollService, ToolbarService, PageService, FilterService, EditService, SortService]
})

export class AppComponent implements OnInit {
  public data: DataManager;
  public editSettings: EditSettingsModel;
  public toolbarOptions: ToolbarItems[];
  public pageSettings: Object ;
  public dataManager: DataManager = new DataManager({
    adaptor: new UrlAdaptor,
    insertUrl: "Home/Insert",
    removeUrl: "Home/Delete",
    updateUrl: "Home/Update",
    url: "Home/DataSource",
  });
};
```



```
ngOnInit(): void {  
    this.data = this.dataManager;  
    this.editSettings = { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Row' };  
    this.toolbarOptions = ['Add', 'Edit', 'Delete', 'Update', 'Cancel'];  
    this.pageSettings = {pageSize: 30};  
}  
}  
`
```

The following code example describes handling of Load on demand at server end.

```
`typescript  
public ActionResult lazyLoading()  
{  
    TreeData.tree = new List<TreeData>();  
    return View();  
}  
  
public ActionResult UrlDatasource(DataManagerRequest dm)  
{  
    List<TreeData> data = new List<TreeData>();  
    data = TreeData.GetTree();  
    DataOperations operation = new DataOperations();  
    IEnumerable<TreeData> DataSource = data;  
    List<TreeData> ExpandedParentRecords = new List<TreeData>();  
    if (dm.Expand != null && dm.Expand[0] == "ExpandingAction") // setting the ExpandStateMapping  
        property whether is true or false  
    {  
        var val = TreeData.GetTree().Where(ds => ds.TaskID == int.Parse(dm.Expand[1])).FirstOrDefault();  
        val.IsExpanded = true;  
    }  
    else if (dm.Expand != null && dm.Expand[0] == "CollapsingAction")  
    {  
        var val = TreeData.GetTree().Where(ds => ds.TaskID == int.Parse(dm.Expand[1])).FirstOrDefault();  
        val.IsExpanded = false;  
    }  
}
```

```
if (!(dm.Where != null && dm.Where.Count > 1))
{
    data = data.Where(p => p.ParentValue == null).ToList();
}
DataSource = data;
if (dm.Search != null && dm.Search.Count > 0) // Searching
{
    DataSource = operation.PerformSearching(DataSource, dm.Search);
}
if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // Sorting
{
    DataSource = operation.PerformSorting(DataSource, dm.Sorted);
}
if (dm.Where != null && dm.Where.Count > 1) //filtering
{
    DataSource = operation.PerformFiltering(DataSource, dm.Where, "and");
}
data = new List<TreeData>();
foreach (var rec in DataSource)
{
    if (rec.IsExpanded)
    {
        ExpandedParentRecords.Add(rec as TreeData); // saving the expanded parent records
    }
    data.Add(rec as TreeData);
}
var GroupData = TreeData.GetTree().ToList().GroupBy(rec => rec.ParentValue)
.Where(g => g.Key != null).ToDictionary(g => g.Key?.ToString(), g => g.ToList());
if (ExpandedParentRecords.Count > 0)
{
    foreach (var Record in ExpandedParentRecords.ToList())
    {
        var ChildGroup = GroupData[Record.TaskID.ToString()];
```

```
if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // sorting the child records
{
    IEnumerable ChildSort = ChildGroup;
    ChildSort = operation.PerformSorting(ChildSort, dm.Sorted);
    ChildGroup = new List<TreeData>();
    foreach (var rec in ChildSort)
    {
        ChildGroup.Add(rec as TreeData);
    }
}

if (dm.Search != null && dm.Search.Count > 0) // searching the child records
{
    IEnumerable ChildSearch = ChildGroup;
    ChildSearch = operation.PerformSearching(ChildSearch, dm.Search);
    ChildGroup = new List<TreeData>();
    foreach (var rec in ChildSearch)
    {
        ChildGroup.Add(rec as TreeData);
    }
}

AppendChildren(dm, ChildGroup, Record, GroupData, data);
}
}

DataSource = data;

if (dm.Expand != null && dm.Expand[0] == "CollapsingAction") // setting the skip index based on
collapsed parent
{
    string IdMapping = "TaskID";
    List<WhereFilter> CollapseFilter = new List<WhereFilter>();
    CollapseFilter.Add(new WhereFilter() { Field = IdMapping, value = dm.Where[0].value, Operator =
dm.Where[0].Operator });
    var CollapsedParentRecord = operation.PerformFiltering(DataSource, CollapseFilter, "and");
    var index = data.Cast<object>().ToList().IndexOf(CollapsedParentRecord.Cast<object>().ToList()[0]);
    dm.Skip = index;
}
```

```
}  
else if (dm.Expand != null && dm.Expand[0] == "ExpandingAction") // setting the skip index based on  
expanded parent  
{  
    string IdMapping = "TaskID";  
    List<WhereFilter> ExpandFilter = new List<WhereFilter>();  
    ExpandFilter.Add(new WhereFilter() { Field = IdMapping, value = dm.Where[0].value, Operator =  
    dm.Where[0].Operator });  
    var ExpandedParentRecord = operation.PerformFiltering(DataSource, ExpandFilter, "and");  
    var index = data.Cast<object>().ToList().IndexOf(ExpandedParentRecord.Cast<object>().ToList()[0]);  
    dm.Skip = index;  
}  
int count = data.Count;  
DataSource = data;  
if (dm.Skip != 0)  
{  
    DataSource = operation.PerformSkip(DataSource, dm.Skip); //Paging  
}  
if (dm.Take != 0)  
{  
    DataSource = operation.PerformTake(DataSource, dm.Take);  
}  
return dm.RequiresCounts ? Json(new { result = DataSource, count = count }) : Json(DataSource);  
}  
  
private void AppendChildren(DataManagerRequest dm, List<TreeData> ChildRecords, TreeData  
ParentValue, Dictionary<string, List<TreeData>> GroupData, List<TreeData> data) // Getting child  
records for the respective parent  
{  
    string TaskId = ParentValue.TaskID.ToString();  
    var index = data.IndexOf(ParentValue);  
    DataOperations operation = new DataOperations();  
    foreach (var Child in ChildRecords)  
    {  
        if (ParentValue.IsExpanded)
```

```
{
string ParentId = Child.ParentValue.ToString();
if (TaskId == ParentId)
{
((IList)data).Insert(++index, Child);
if (GroupData.ContainsKey(Child.TaskID.ToString()))
{
var DeepChildRecords = GroupData[Child.TaskID.ToString()];
if (DeepChildRecords?.Count > 0)
{
if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // sorting the child records
{
IEnumerable ChildSort = DeepChildRecords;
ChildSort = operation.PerformSorting(ChildSort, dm.Sorted);
DeepChildRecords = new List<TreeData>();
foreach (var rec in ChildSort)
{
DeepChildRecords.Add(rec as TreeData);
}
}
if (dm.Search != null && dm.Search.Count > 0) // searching the child records
{
IEnumerable ChildSearch = DeepChildRecords;
ChildSearch = operation.PerformSearching(ChildSearch, dm.Search);
DeepChildRecords = new List<TreeData>();
foreach (var rec in ChildSearch)
{
DeepChildRecords.Add(rec as TreeData);
}
}
AppendChildren(dm, DeepChildRecords, Child, GroupData, data);
if (Child.IsExpanded)
{

```

```
index += DeepChildRecords.Count;
}
}
}
else
{
    Child.isParent = false;
}
}
}
}
}
}
public ActionResult Update(CRUDModel<TreeData> value)
{
    List<TreeData> data = new List<TreeData>();
    data = TreeData.GetTree();
    var val = data.Where(ds => ds.TaskID == value.Value.TaskID).FirstOrDefault();
    val.TaskName = value.Value.TaskName;
    val.Duration = value.Value.Duration;
    return Json(val);
}
public ActionResult Insert(CRUDModel<TreeData> value)
{
    var c = 0;
    for (; c < TreeData.GetTree().Count; c++)
    {
        if (TreeData.GetTree()[c].TaskID == value.RelationalKey)
        {
            if (TreeData.GetTree()[c].isParent == null)
            {
                TreeData.GetTree()[c].isParent = true;
            }
            break;
        }
    }
}
```

```
}  
}  
c += FindChildRecords(value.RelationalKey);  
TreeData.GetTree().Insert(c + 1, value.Value);  
return Json(value.Value);  
}  
public int FindChildRecords(int? id)  
{  
    var count = 0;  
    for (var i = 0; i < TreeData.GetTree().Count; i++)  
    {  
        if (TreeData.GetTree()[i].ParentValue == id)  
        {  
            count++;  
            count += FindChildRecords(TreeData.GetTree()[i].TaskID);  
        }  
    }  
    return count;  
}  
public object Delete(CRUDModel<TreeData> value)  
{  
    if (value.deleted != null)  
    {  
        for (var i = 0; i < value.deleted.Count; i++)  
        {  
            TreeData.GetTree().Remove(TreeData.GetTree().Where(ds => ds.TaskID ==  
            value.deleted[i].TaskID).FirstOrDefault());  
        }  
    }  
    else  
    {  
        TreeData.GetTree().Remove(TreeData.GetTree().Where(or => or.TaskID ==  
        int.Parse(value.Key.ToString())).FirstOrDefault());  
    }  
}
```

```
return Json(value);
}

public class CRUDModel<T> where T : class
{
    public TreeData Value;
    public int Key { get; set; }
    public int RelationalKey { get; set; }
    public List<T> added { get; set; }
    public List<T> changed { get; set; }
    public List<T> deleted { get; set; }
}

public class TreeData
{
    public static List<TreeData> tree = new List<TreeData>();
    [System.ComponentModel.DataAnnotations.Key]
    public int TaskID { get; set; }
    public string TaskName { get; set; }
    public int Duration { get; set; }
    public int? ParentValue { get; set; }
    public bool? isParent { get; set; }
    public bool IsExpanded { get; set; }
    public TreeData() { }
    public static List<TreeData> GetTree()
    {
        if (tree.Count == 0)
        {
            int root = 0;
            for (var t = 1; t <= 500; t++)
            {
                Random ran = new Random();
                string math = (ran.Next() % 3) == 0 ? "High" : (ran.Next() % 2) == 0 ? "Release Breaker" : "Critical";
                string progr = (ran.Next() % 3) == 0 ? "Started" : (ran.Next() % 2) == 0 ? "Open" : "In Progress";
                root++;
            }
        }
    }
}
```



```

int rootItem = root;

tree.Add(new TreeData() { TaskID = rootItem, TaskName = "Parent task " + rootItem.ToString(), isParent
= true, IsExpanded = false, ParentValue = null, Duration = ran.Next(1, 50) });

int parent = root;
for (var d = 0; d < 1; d++)
{
    root++;
    string value = ((parent + 1) % 3 == 0) ? "Low" : "Critical";
    int par = parent + 1;
    progr = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";
    int iD = root;

    tree.Add(new TreeData() { TaskID = iD, TaskName = "Child task " + iD.ToString(), isParent = true,
    IsExpanded = false, ParentValue = rootItem, Duration = ran.Next(1, 50) });

    int subparent = root;
    for (var c = 0; c < 500; c++)
    {
        root++;
        string val = ((subparent + c + 1) % 3 == 0) ? "Low" : "Critical";
        int subchild = subparent + c + 1;
        string progress = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";
        int childID = root ;

        tree.Add(new TreeData() { TaskID = childID, TaskName = "sub Child task " + childID.ToString(), isParent =
        false, IsExpanded = false, ParentValue = subparent, Duration = ran.Next(1, 50) });
    }
}
return tree;
}
}
,

```

[Load parent rows in expanded state with virtualization](#)

Tree Grid provides an option to load the child records in the initial rendering itself for remote data binding by setting the [loadChildOnDemand](#) as true. When the [loadChildOnDemand](#) is enabled, parent records are rendered in expanded state.

When using virtualization with `loadChildOnDemand`, it helps you to improve the tree grid performance while loading the child records during the initial rendering for remote data binding by setting [enableVirtualization](#) as true and `loadChildOnDemand` as true.

```
`typescript

import { Component, OnInit } from '@angular/core';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
import { EditSettingsModel, ToolbarItems, VirtualScrollService, ToolbarService, PageService,
FilterService, EditService, SortService } from '@syncfusion/ej2-angular-treegrid';

@Component({
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1' height='400' idMapping='TaskID'
  parentIdMapping='ParentValue' hasChildMapping='isParent' expandStateMapping='IsExpanded'
  loadChildOnDemand='true' enableVirtualization='true' allowFiltering='true' allowSorting='true'
  [allowPaging]='true' [pageSettings]='pageSettings' [toolbar]='toolbarOptions'
  [editSettings]='editSettings'>
  <e-columns>
  <e-column field='TaskID' headerText='Task ID' width='90' textAlign='Right'></e-column>
  <e-column field='TaskName' headerText='Task Name' width='180'></e-column>
  <e-column field='Duration' headerText='Duration' width='80' textAlign='Right'></e-column>
  </e-columns>
  </ejs-treegrid>`,
  providers: [VirtualScrollService, ToolbarService, PageService, FilterService, EditService, SortService]
})

export class AppComponent implements OnInit {
  public data: DataManager;
  public editSettings: EditSettingsModel;
  public toolbarOptions: ToolbarItems[];
  public pageSettings: Object ;
  public dataManager: DataManager = new DataManager({
    adaptor: new UrlAdaptor,
    insertUrl: "Home/Insert",
    removeUrl: "Home/Delete",
    updateUrl: "Home/Update",
    url: "Home/DataSource",
  });
}
```

```

ngOnInit(): void {
  this.data = this.dataManager;
  this.editSettings = { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Row' };
  this.toolbarOptions = ['Add', 'Edit', 'Delete', 'Update', 'Cancel'];
  this.pageSettings = { pageSize: 30 };
}
}
`

```

The following code example describes handling of child records at server end.

```

`typescript
public ActionResult loadChildOndemand()
{
  TreeData.tree = new List<TreeData>();
  return View();
}

public ActionResult UrlDatasource(DataManagerRequest dm)
{
  List<TreeData> data = new List<TreeData>();
  data = TreeData.GetTree();
  DataOperations operation = new DataOperations();
  IEnumerable<TreeData> DataSource = data;
  if (dm.Expand != null && dm.Expand[0] == "CollapsingAction") // setting the ExpandStateMapping
  property whether is true or false
  {
    var val = TreeData.GetTree().Where(ds => ds.TaskID == int.Parse(dm.Expand[1])).FirstOrDefault();
    val.IsExpanded = false;
  }
  else if (dm.Expand != null && dm.Expand[0] == "ExpandingAction")
  {
    var val = TreeData.GetTree().Where(ds => ds.TaskID == int.Parse(dm.Expand[1])).FirstOrDefault();
    val.IsExpanded = true;
  }
  if (!(dm.Where != null && dm.Where.Count > 1))

```

```
{
data = data.Where(p => p.ParentValue == null).ToList();
}
DataSource = data;
if (dm.Search != null && dm.Search.Count > 0) // Searching
{
DataSource = operation.PerformSearching(DataSource, dm.Search);
}
if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // Sorting
{
DataSource = operation.PerformSorting(DataSource, dm.Sorted);
}
if (dm.Where != null && dm.Where.Count > 1) //filtering
{
DataSource = operation.PerformFiltering(DataSource, dm.Where, "and");
}
data = new List<TreeData>();
foreach (var rec in DataSource)
{
data.Add(rec as TreeData);
}
var GroupData = TreeData.GetTree().ToList().GroupBy(rec => rec.ParentValue)
.Where(g => g.Key != null).ToDictionary(g => g.Key?.ToString(), g => g.ToList());
foreach (var Record in data.ToList())
{
if (GroupData.ContainsKey(Record.TaskID.ToString()))
{
var ChildGroup = GroupData[Record.TaskID.ToString()];
if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // Sorting the child records
{
IEnumerable ChildSort = ChildGroup;
ChildSort = operation.PerformSorting(ChildSort, dm.Sorted);
ChildGroup = new List<TreeData>();
}
```

```
foreach (var rec in ChildSort)
{
    ChildGroup.Add(rec as TreeData);
}
}

if (dm.Search != null && dm.Search.Count > 0) // Searching the child records
{
    IEnumerable ChildSearch = ChildGroup;
    ChildSearch = operation.PerformSearching(ChildSearch, dm.Search);
    ChildGroup = new List<TreeData>();
    foreach (var rec in ChildSearch)
    {
        ChildGroup.Add(rec as TreeData);
    }
}

if (ChildGroup?.Count > 0)
AppendChildren(dm, ChildGroup, Record, GroupData, data);
}
}

DataSource = data;

if (dm.Expand != null && dm.Expand[0] == "CollapsingAction") // setting the skip index based on
collapsed parent
{
    string IdMapping = "TaskID";
    List<WhereFilter> CollapseFilter = new List<WhereFilter>();
    CollapseFilter.Add(new WhereFilter() { Field = IdMapping, value = dm.Where[0].value, Operator =
dm.Where[0].Operator });
    var CollapsedParentRecord = operation.PerformFiltering(DataSource, CollapseFilter, "and");
    var index = data.Cast<object>().ToList().IndexOf(CollapsedParentRecord.Cast<object>().ToList()[0]);
    dm.Skip = index;
}

else if (dm.Expand != null && dm.Expand[0] == "ExpandingAction") // setting the skip index based on
expanded parent
{

```

```
string IdMapping = "TaskID";
List<WhereFilter> ExpandFilter = new List<WhereFilter>();
ExpandFilter.Add(new WhereFilter() { Field = IdMapping, value = dm.Where[0].value, Operator =
dm.Where[0].Operator });
var ExpandedParentRecord = operation.PerformFiltering(DataSource, ExpandFilter, "and");
var index = data.Cast<object>().ToList().IndexOf(ExpandedParentRecord.Cast<object>().ToList()[0]);
dm.Skip = index;
}
int count = data.Count;
DataSource = data;
if (dm.Skip != 0)
{
DataSource = operation.PerformSkip(DataSource, dm.Skip); //Paging
}
if (dm.Take != 0)
{
DataSource = operation.PerformTake(DataSource, dm.Take);
}
return dm.RequiresCounts ? Json(new { result = DataSource, count = count }) : Json(DataSource);
}

private void AppendChildren(DataManagerRequest dm, List<TreeData> ChildRecords, TreeData
ParentValue, Dictionary<string, List<TreeData>> GroupData, List<TreeData> data) // Getting child
records for the respective parent
{
string TaskId = ParentValue.TaskID.ToString();
var index = data.IndexOf(ParentValue);
DataOperations operation = new DataOperations();
foreach (var Child in ChildRecords)
{
if (ParentValue.IsExpanded)
{
string ParentId = Child.ParentValue.ToString();
if (TaskId == ParentId)
{

```

```
((IList)data).Insert(++index, Child);
if (GroupData.ContainsKey(Child.TaskID.ToString()))
{
    var DeepChildRecords = GroupData[Child.TaskID.ToString()];
    if (DeepChildRecords?.Count > 0)
    {
        if (dm.Sorted != null && dm.Sorted.Count > 0 && dm.Sorted[0].Name != null) // sorting the child records
        {
            IEnumerable ChildSort = DeepChildRecords;
            ChildSort = operation.PerformSorting(ChildSort, dm.Sorted);
            DeepChildRecords = new List<TreeData>();
            foreach (var rec in ChildSort)
            {
                DeepChildRecords.Add(rec as TreeData);
            }
        }
        if (dm.Search != null && dm.Search.Count > 0) // searching the child records
        {
            IEnumerable ChildSearch = DeepChildRecords;
            ChildSearch = operation.PerformSearching(ChildSearch, dm.Search);
            DeepChildRecords = new List<TreeData>();
            foreach (var rec in ChildSearch)
            {
                DeepChildRecords.Add(rec as TreeData);
            }
        }
        AppendChildren(dm, DeepChildRecords, Child, GroupData, data);
        if (Child.IsExpanded)
        {
            index += DeepChildRecords.Count;
        }
    }
}
```

```
}  
}  
}  
}  
public ActionResult Update(CRUDModel<TreeData> value)  
{  
    List<TreeData> data = new List<TreeData>();  
    data = TreeData.GetTree();  
    var val = data.Where(ds => ds.TaskID == value.Value.TaskID).FirstOrDefault();  
    val.TaskName = value.Value.TaskName;  
    val.Duration = value.Value.Duration;  
    return Json(val);  
}  
public ActionResult Insert(CRUDModel<TreeData> value)  
{  
    var c = 0;  
    for (; c < TreeData.GetTree().Count; c++)  
    {  
        if (TreeData.GetTree()[c].TaskID == value.RelationalKey)  
        {  
            if (TreeData.GetTree()[c].isParent == null)  
            {  
                TreeData.GetTree()[c].isParent = true;  
            }  
            break;  
        }  
    }  
    c += FindChildRecords(value.RelationalKey);  
    TreeData.GetTree().Insert(c + 1, value.Value);  
    return Json(value.Value);  
}  
public int FindChildRecords(int? id)  
{
```



```
var count = 0;
for (var i = 0; i < TreeData.GetTree().Count; i++)
{
    if (TreeData.GetTree()[i].ParentValue == id)
    {
        count++;
        count += FindChildRecords(TreeData.GetTree()[i].TaskID);
    }
}
return count;
}

public object Delete(CRUDModel<TreeData> value)
{
    if (value.deleted != null)
    {
        for (var i = 0; i < value.deleted.Count; i++)
        {
            TreeData.GetTree().Remove(TreeData.GetTree().Where(ds => ds.TaskID ==
            value.deleted[i].TaskID).FirstOrDefault());
        }
    }
    else
    {
        TreeData.GetTree().Remove(TreeData.GetTree().Where(or => or.TaskID ==
        int.Parse(value.Key.ToString())).FirstOrDefault());
    }
    return Json(value);
}

public class CRUDModel<T> where T : class
{
    public TreeData Value;
    public int Key { get; set; }
    public int RelationalKey { get; set; }
    public List<T> added { get; set; }
```

```
public List<T> changed { get; set; }
public List<T> deleted { get; set; }
}

public class TreeData
{
    public static List<TreeData> tree = new List<TreeData>();
    [System.ComponentModel.DataAnnotations.Key]
    public int TaskID { get; set; }
    public string TaskName { get; set; }
    public int Duration { get; set; }
    public int? ParentValue { get; set; }
    public bool? isParent { get; set; }
    public bool IsExpanded { get; set; }
    public TreeData() { }
    public static List<TreeData> GetTree()
    {
        if (tree.Count == 0)
        {
            int root = 0;
            for (var t = 1; t <= 1500; t++)
            {
                Random ran = new Random();
                string math = (ran.Next() % 3) == 0 ? "High" : (ran.Next() % 2) == 0 ? "Release Breaker" : "Critical";
                string progr = (ran.Next() % 3) == 0 ? "Started" : (ran.Next() % 2) == 0 ? "Open" : "In Progress";
                root++;
                int rootItem = root;
                tree.Add(new TreeData() { TaskID = rootItem, TaskName = "Parent task " + rootItem.ToString(), isParent
                = true, IsExpanded = true, ParentValue = null, Duration = ran.Next(1, 50) });
                int parent = root;
                for (var d = 0; d < 1; d++)
                {
                    root++;
                    string value = ((parent + 1) % 3 == 0) ? "Low" : "Critical";
```

```

int par = parent + 1;
progr = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";
int iD = root;
tree.Add(new TreeData() { TaskID = iD, TaskName = "Child task " + iD.ToString(), isParent = true,
IsExpanded = true, ParentValue = rootItem, Duration = ran.Next(1, 50) });
int subparent = root;
for (var c = 0; c < 6; c++)
{
root++;
string val = ((subparent + c + 1) % 3 == 0) ? "Low" : "Critical";
int subchild = subparent + c + 1;
string progress = (ran.Next() % 3) == 0 ? "In Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";
int childID = root ;
tree.Add(new TreeData() { TaskID = childID, TaskName = "sub Child task " + childID.ToString(),
ParentValue = subparent, Duration = ran.Next(1, 50) });
}
}
}
}
return tree;
}
}
、

```

Immutable mode in Angular Treegrid component

The immutable mode optimizes the Tree Grid re-rendering performance by using the object reference and [deep compare](#) concept. When performing the Tree Grid actions, it will only re-render the modified or newly added rows and prevent the re-rendering of the unchanged rows.

To enable this feature, you have to set the [enableImmutableMode](#) property as **true**.

* This feature uses the primary key value for data comparison. So, you need to provide the [isPrimaryKey](#) column.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, EditService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'

```

```

import { Component, OnInit, ViewChild } from "@angular/core";
import { ButtonComponent } from "@syncfusion/ej2-angular-buttons";
import { GridComponent } from '@syncfusion/ej2-angular-grids';
import { sampleData2, dataSource1, sampleData } from "../datasource";
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService, EditService],
  standalone: true,
  selector: 'app-container',
  templateUrl: './app.template.html'
})
export class AppComponent implements OnInit {
  public data: Object[] = [];
  public pageSettings: Object = { pageSize: 50 };
  public editSettings: Object = { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Cell' };
  public selectionOptions: Object = { type: 'Multiple' };
  public immutableStart?: number;
  public normalStart?: number;
  @ViewChild("immutable")
  public immutableGrid?: GridComponent;
  @ViewChild("normal")
  public normalGrid?: GridComponent;
  @ViewChild("addtop")
  public addtop?: ButtonComponent;
  @ViewChild("addbottom")
  public addbottom?: ButtonComponent;
  @ViewChild("delete")
  public delete?: ButtonComponent;
  @ViewChild("reverse")
  public reverse?: ButtonComponent;
  @ViewChild("paging")
  public paging?: ButtonComponent;
  public immutableInit: boolean = true;
  public init: boolean = true;
  ngOnInit(): void {
    dataSource1();
    this.data = sampleData2;
  }
  immutableBeforeDataBound(args: any): void {
    this.immutableStart = new Date().getTime();
  }
  immutableDataBound(args: any): void {
    let val: number | string = this.immutableInit ? '' : new
Date().getTime() - (this.immutableStart as number);
    (document.getElementById("immutableDelete") as HTMLElement).innerHTML =
      "Immutable rendering Time: " + "<b>" + val + "</b>" + "<b>ms</b>";
    this.immutableStart = 0; this.immutableInit = false;
  }
  normalBeforeDataBound(args: any): void {
    this.normalStart = new Date().getTime();
  }
}

```

```

normalDataBound(args: any): void {
    let val: number = (this.init ? '' : new Date().getTime() as number -
    (this.normalStart as number)) as number;
    (document.getElementById("normalDelete") as HTMLElement).innerHTML =
    "Normal rendering Time: " + "<b>" + val + "</b>" + "<b>ms</b>";
    this.normalStart = 0; this.init = false;
}
addTopEvent(): void {
    (this.normalGrid as GridComponent).addRecord(sampleData[0], 0);
    (this.immutableGrid as GridComponent).addRecord(sampleData[0], 0);
}
addBottomEvent(): void {
    (this.normalGrid as GridComponent).addRecord(sampleData[0], 0);
    (this.immutableGrid as GridComponent).addRecord(sampleData[0], 0);
}
deleteEvent(): void {
    (this.normalGrid as GridComponent).selectRows([0, 2, 4]);
    (this.immutableGrid as GridComponent).selectRows([0, 2, 4]);
    (this.normalGrid as GridComponent).deleteRecord();
    (this.immutableGrid as GridComponent).deleteRecord();
}
sortEvent(): void {
    let aData: object[] = ((this.immutableGrid as GridComponent).dataSource
as object[]).reverse();
    (this.normalGrid as GridComponent).setProperties({ dataSource: aData });
    (this.immutableGrid as GridComponent).setProperties({ dataSource: aData
});
}
pageEvent(): void {
    let page: number = ((this.normalGrid as
GridComponent).pageSettings.currentPage as number + 1) as number;
    (this.normalGrid as GridComponent).goToPage(page);
    (this.immutableGrid as GridComponent).goToPage(page);
}
}

```

MAIN.TS

```

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

```

Limitations

The following features are not supported in the immutable mode:

- Frozen rows and columns
- Row Template
- Detail Template
- Column reorder
- Virtualization

You can refer to our [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Observables in Angular Treegrid component

An [Link to the Video](#) is used extensively by Angular since it provides significant benefits over techniques for event handling, asynchronous programming, and handling multiple values.

You can also check on this video for Observable binding in Tree Grid:

Observable binding using Async pipe

TreeGrid 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.

Data binding

The TreeGrid expects an object from the **Observable**. The emitted value should be an object with properties **result** and **count**.

```
import { Component, OnInit, ViewChild } from '@angular/core';
import { Observable } from 'rxjs/Observable';
import { CrudService } from './data.service';
import { Tasks } from './tasks';
import { TreeGridComponent, DataStateChangeEventArgs } from '@syncfusion/ej2-angular-treegrid';

@Component({
  selector: 'app-root',
  template: `<ejs-treegrid #treegrid [dataSource]='data | async' [treeColumnIndex]='1'
  parentIdMapping='ParentId' idMapping='TaskId' hasChildMapping='isParent' (dataStateChange)=
  'dataStateChange($event)' [allowPaging]="true" [allowSorting]="true" [pageSettings]="pageSettings">
  <e-columns>
  <e-column field='TaskId' headerText='Task ID' width='90' textAlign='Right'></e-column>
  <e-column field='TaskName' headerText='Task Name' width='170'></e-column>
  <e-column field='Progress' headerText='Progress' width='130' textAlign='Right'></e-column>
  <e-column field='Duration' headerText='Duration' width='80' textAlign='Right'></e-column>
  </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data: Observable<DataStateChangeEventArgs>;
  public pageSettings: Object;
  public state: DataStateChangeEventArgs;
```

```

@ViewChild('treegrid')
public treegrid: TreeGridComponent;
tasks: Tasks[];
constructor(private dataService: DataService) {
  this.data = dataService;
}
public dataStateChange(state: DataStateChangeEventArgs): void {
  this.dataService.execute(state);
}
public ngOnInit(): void {
  this.pageSettings = { pageSize: 1, pageSizeMode: 'Root' };
  const state: any = { skip: 0, take: 1 };
  this.dataService.execute(state);
}
,
,

import { DataManager, Query, } from '@syncfusion/ej2-data';
import { Http } from '@angular/http';
import { Injectable } from '@angular/core';
import { DataStateChangeEventArgs } from '@syncfusion/ej2-angular-treegrid';
import { Subject } from 'rxjs/Subject';
import { Observable } from 'rxjs/Observable';
@Injectable()
export class DataService extends Subject<Object> {
  private BASE_URL = '/api/tasks'; //// provide the service url required to fetch data from server
  constructor(private http: Http) {
    super();
  }
  public getData(state: DataStateChangeEventArgs): Observable<DataStateChangeEventArgs> {
    const pageQuery = `?skip=${state.skip}&stop=${state.take}`;
    /// filter query for fetching only the root level records
    const treegridQuery = `?filter='ParentId eq null'`;

```

```

return this.http
.get(`${this.BASE_URL}?${pageQuery}&${treegridQuery}&$inlinecount=allpages&$format=json`)
.map((response: any) => response.json())
.map((response: any) => (<DataResult>{
result: response['d']['results'],
count: parseInt(response['d']['count'], 10)
})))
.map((data: any) => data);
}

public execute(state: DataStateChangeEventArgs): void {
this.getData(state).subscribe(x => {
super.next(x)
});
}
}
`

```

You should maintain the same `Observable` instance for every treegrid actions.

We have a limitation for Custom Binding feature of TreeGrid. This feature works only for Self Referential data binding with `pageSizeMode` as Root.

Handling child data

Using the custom binding feature you can bind the child data for a parent record as per your custom logic. When a parent record is expanded, [dataStateChange](#) event is triggered in which you can assign your custom data to the `childData` property of the [dataStateChange](#) event arguments.

After assigning the child data, `childDataBind` method should be called from the [dataStateChange](#) event arguments to indicate that the data is bound.

```

import { Component, OnInit, ViewChild } from '@angular/core';
import { Observable } from 'rxjs/Observable';
import { CrudService } from './data.service';
import { Tasks } from './tasks';
import { TreeGridComponent, DataStateChangeEventArgs } from '@syncfusion/ej2-angular-treegrid';
@Component({
selector: 'app-root',

```



```
template: `<ejs-treegrid #treegrid [dataSource]='data | async' [treeColumnIndex]='1'
parentIdMapping='ParentId' idMapping='TaskId' hasChildMapping='isParent' height=265
(dataStateChange)= 'dataStateChange($event)' [allowPaging]="true" [allowSorting]="true"
[pageSettings]="pageSettings">
<e-columns>
<e-column field='TaskId' headerText='Task ID' width='90' textAlign='Right'></e-column>
<e-column field='TaskName' headerText='Task Name' width='170'></e-column>
<e-column field='Progress' headerText='Progress' width='130' textAlign='Right'></e-column>
<e-column field='Duration' headerText='Duration' width='80' textAlign='Right'></e-column>
</e-columns>
</ejs-treegrid>`
})
export class AppComponent implements OnInit {
public data: Observable<DataStateChangeEventArgs>;
public pageSettings: Object;
public state: DataStateChangeEventArgs;
@ViewChild('treegrid')
public treegrid: TreeGridComponent;
tasks: Tasks[];
constructor(private dataService: DataService) {
this.data = dataService;
}
public dataStateChange(state: DataStateChangeEventArgs): void {
if (state.requestType === 'expand') {
///// assigning the child data for the expanded record.
state.childData = <any>[
{ TaskId: 2, TaskName: 'VINER', ParentId: 1, Duration: 23, Progress: 34, isParent: false },
{ TaskId: 3, TaskName: 'JOHN', ParentId: 1, Duration: 23, Progress: 34, isParent: false },
{ TaskId: 4, TaskName: 'TOMSP', ParentId: 1, Duration: 23, Progress: 34, isParent: false }
]
state.childDataBind();    ///// to indicate that the child data is bound.
} else {
this.dataService.execute(state);
}
```

```

}
public ngOnInit(): void {
  this.pageSettings = { pageSize: 1, pageSizeMode: 'Root' };
  const state: any = { skip: 0, take: 1 };
  this.dataService.execute(state);
}
}
,
,

import { DataManager, Query, } from '@syncfusion/ej2-data';
import { Http } from '@angular/http';
import { Injectable } from '@angular/core';
import { DataStateChangeEventArgs } from '@syncfusion/ej2-angular-treegrid';
import { Subject } from 'rxjs/Subject';
import { Observable } from 'rxjs/Observable';
@Injectable()
export class DataService extends Subject<Object> {
  private BASE_URL = '/api/tasks'; //// provide the service url required to fetch data from server
  private dataManager = new DataManager({
    url: this.BASE_URL,
    crossDomain: true
  });
  constructor(private http: Http) {
    super();
  }
  public getData(state: DataStateChangeEventArgs): Observable<DataStateChangeEventArgs> {
    const pageQuery = `${skip}=${state.skip}&${top}=${state.take}`;
    /// filter query for fetching only the root level records
    const treegridQuery = `${filter}='ParentId eq null'`;
    return this.http
      .get(`${this.BASE_URL}?${pageQuery}&${treegridQuery}&${inlinecount}=allpages&${format}=json`)
      .map((response: any) => response.json())
      .map((response: any) => (<DataResult>{

```

```

result: response['d']['results'],
count: parseInt(response['d']['count'], 10)
)))
.map((data: any) => data);
}
public execute(state: DataStateChangeEventArgs): void {
this.getData(state).subscribe(x => {
super.next(x)
});
}
}
,

```

Handling TreeGrid actions

For TreeGrid actions such as **paging**, **sorting**, etc., the **dataStateChange** event is invoked. You have to query and resolve data using **Observable** in this event based on the state arguments.

```

import { Component, OnInit, Inject, ViewChild } from '@angular/core';
import { TreeGridComponent, DataStateChangeEventArgs } from '@syncfusion/ej2-angular-treegrid';
import { DataService } from './data.service';

@Component({
selector: 'app-root',

templateUrl: `<ejs-treegrid #treegrid [dataSource]='data | async' [treeColumnIndex]='1'
parentIdMapping='ParentId' idMapping='TaskId' hasChildMapping='isParent' (dataStateChange)=
'dataStateChange($event)' [allowPaging]="true" [allowSorting]="true" [pageSettings]="pageSettings">
<e-columns>
<e-column field='TaskId' headerText='Task ID' width='90' textAlign='Right'></e-column>
<e-column field='TaskName' headerText='Task Name' width='170'></e-column>
<e-column field='Progress' headerText='Progress' width='130' textAlign='Right'></e-column>
<e-column field='Duration' headerText='Duration' width='80' textAlign='Right'></e-column>
</e-columns>
</ejs-treegrid>`
})
export class AppComponent implements OnInit {
title = 'app';

```

```

@ViewChild('treegrid') private treegrid: TreeGridComponent;
public data: Object;
public state: DataStateChangeEventArgs;
public pageSettings: any;
constructor(@Inject(DataService) private service: DataService) {
  this.data = service;
}
public dataStateChange(state: DataStateChangeEventArgs): void {
  this.service.execute(state);
}
ngOnInit() {
  this.pageSettings = { pageCount: 4 };
  let state = { skip: 0, take: 12 };
  this.service.execute(state);
}
}
`

```

When initial rendering, the `dataStateChange` event will not be triggered. You can perform the operation in the `ngOnInit` if you want the treegrid to show the record.

Perform CRUD operations

The [dataSourceChanged](#) event is triggered to update the treegrid data. You can perform the save operation based on the event arguments, and you need to call the [endEdit](#) method to indicate the completion of save operation.

```

import { Component, OnInit, ViewChild } from '@angular/core';
import { Observable } from 'rxjs/Observable';
import { CrudService } from './crud.service';
import { Tasks } from './tasks';
import { DataSourceChangedEventArgs } from '@syncfusion/ej2-grids';
import { DataStateChangeEventArgs, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  selector: 'app-root',
  template: `<ejs-treegrid #treegrid [dataSource]='data | async' [treeColumnIndex]='1'
  parentIdMapping='ParentId' idMapping='TaskId' hasChildMapping='isParent'
  (dataSourceChanged)='dataSourceChanged($event)' (dataStateChange)= 'dataStateChange($event)'

```

```
[allowPaging]="true" [allowSorting]="true" [pageSettings]="pageSettings" [editSettings]="editSettings"
[toolbar]="toolbar">
<e-columns>
<e-column field='TaskId' headerText='Task ID' width='90' textAlign='Right'></e-column>
<e-column field='TaskName' headerText='Task Name' width='170'></e-column>
<e-column field='Progress' headerText='Progress' width='130' textAlign='Right'></e-column>
<e-column field='Duration' headerText='Duration' width='80' textAlign='Right'></e-column>
</e-columns>
</ejs-treegrid>`
})
export class AppComponent implements OnInit {
public data: Observable<DataStateChangeEventArgs>;
public pageOptions: Object;
public editSettings: Object;
public pageSettings: Object;
public toolbar: string[];
public state: DataStateChangeEventArgs;
@ViewChild('treegrid')
public treegrid: TreeGridComponent;
tasks: Tasks[];
constructor(private crudService: CrudService) {
this.data = crudService;
}
public dataStateChange(state: DataStateChangeEventArgs): void {
this.crudService.execute(state);
}
public dataSourceChanged(state: DataSourceChangedEventArgs): void {
if (state.action === 'add') {
this.crudService.addRecord(state).subscribe(() => {
state.endEdit()
});
this.crudService.addRecord(state).subscribe(() => { }, error => console.log(error), () => {
state.endEdit()
}
```

```
});
} else if (state.action === 'edit') {
this.crudService.updateRecord(state).subscribe(() => {
state.endEdit()
}
);
} else if (state.requestType === 'delete') {
this.crudService.deleteRecord(state).subscribe(() => {
state.endEdit()
});
}
}

public ngOnInit(): void {
this.editSettings = { allowEditing: true, allowAdding: true, allowDeleting: true, mode: 'Normal' };
this.toolbar = ['Add', 'Edit', 'Delete', 'Update', 'Cancel'];
this.pageSettings = { pageSize: 1, pageSizeMode: 'Root' };
const state: any = { skip: 0, take: 1 };
this.crudService.execute(state);
}
}
,
,

import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable } from 'rxjs/Observable';
import 'rxjs/add/operator/map';
import { Subject } from 'rxjs/Subject';
import { Tasks } from './tasks';
import { DataStateChangeEventArgs, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { DataSourceChangedEventArgs } from '@syncfusion/ej2-grids';
const httpOptions = {
headers: new HttpHeaders({ 'Content-Type': 'application/json' })
};
```

```
@Injectable()
export class CrudService extends Subject<DataStateChangeEventArgs> {
  private tasksUrl = 'api/tasks'; // URL to web api for fetching the data
  constructor(
    private http: HttpClient) {
    super();
  }
  public execute(state: any): void {
    if (state.requestType === 'expand') {
      this.getChildData(state).subscribe(x => super.next(x as DataStateChangeEventArgs));
    } else {
      this.getAllData(state).subscribe(x => super.next(x as DataStateChangeEventArgs));
    }
  }
  / GET all data from the server */
  getAllData( state?: any): Observable<any[]> {
    return this.http.get<Tasks[]>(this.tasksUrl)
      .map((response: any) => (<any>{
        result: state.take > 0 ? response.slice(0, state.take) : response,
        count: 2
      })))
  }
  getChildData( state?: any): Observable<any[]> {
    return this.http.get<Tasks[]>(this.tasksUrl)
      .map((response: any) => (<any>{
        result : response.slice(1, 3),
      })))
  }
  / POST: add a new record to the server */
  addRecord(state: DataSourceChangedEventArgs): Observable<Tasks> {
    // you can apply empty string instead of state.data to get failure(error)
    return this.http.post<Tasks>(this.tasksUrl, state.data, httpOptions);
  }
}
```

```

/ DELETE: delete the record from the server */
deleteRecord(state: any): Observable<Tasks> {
  const id = state.data[0].id;
  const url = `${this.tasksUrl}/${id}`;
  return this.http.delete<Tasks>(url, httpOptions);
}

/ PUT: update the record on the server */
updateRecord(state: DataSourceChangedEventArgs): Observable<any> {
  return this.http.put(this.tasksUrl, state.data, httpOptions);
}
}
,

```

Calculate aggregates

The footer aggregate values should be calculated and sent along with the `dataSource` property as follows. The aggregate property of the data source should contain the aggregate value assigned to the `field – type` property. For example, the `Sum` aggregate value for the `Duration` field should be assigned to the `Duration - sum` property.

```

`json
{
  result: [{..}, {..}, {..}, ...],
  count: 830,
  aggregates: { 'Duration - sum' : 450 }
}
,

```

The group footer and caption aggregate values can be calculated by the treegrid itself.

Provide Excel Filter data source

The `dataStateChange` event is triggered with appropriate arguments when the Excel filter requests the filter choice data source. You need to resolve the Excel filter data source using the `dataSource` resolver function from the state argument as follows.

```

,

import { Component, OnInit, ViewChild } from '@angular/core';
import { Observable } from 'rxjs/Observable';
import { CrudService } from './data.service';
import { Tasks } from './tasks';
import { TreeGridComponent, DataStateChangeEventArgs } from '@syncfusion/ej2-angular-treegrid';

```



```
@Component({
  selector: 'app-root',
  template: `<ejs-treegrid #treegrid [dataSource]='data | async' [treeColumnIndex]='1'
  parentIdMapping='ParentId' idMapping='TaskId' hasChildMapping='isParent' (dataStateChange)=
  'dataStateChange($event)' [allowPaging]="true" [allowSorting]="true" [pageSettings]="pageSettings">
  <e-columns>
  <e-column field='TaskId' headerText='Task ID' width='90' textAlign='Right'></e-column>
  <e-column field='TaskName' headerText='Task Name' width='170'></e-column>
  <e-column field='Progress' headerText='Progress' width='130' textAlign='Right'></e-column>
  <e-column field='Duration' headerText='Duration' width='80' textAlign='Right'></e-column>
  </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data: Observable<DataStateChangeEventArgs>;
  public pageSettings: Object;
  public state: DataStateChangeEventArgs;
  @ViewChild('treegrid')
  public treegrid: TreeGridComponent;
  tasks: Tasks[];
  constructor(private dataService: DataService) {
    this.data = dataService;
  }
  public dataStateChange(state: DataStateChangeEventArgs): void {
    if (state.action.requestType === 'filterchoicerequest' || state.action.requestType === 'filtersearchbegin')
    {
      this.service.getData(state).subscribe((e) => state.dataSource(e));
    } else {
      this.service.execute(state);
    }
  }
  public ngOnInit(): void {
    this.pageSettings = { pageSize: 1, pageSizeMode: 'Root' };
    const state: any = { skip: 0, take: 1 };
  }
}
```

```
this.dataService.execute(state);
}
}
,
,

import { DataManager, Query, } from '@syncfusion/ej2-data';
import { Http } from '@angular/http';
import { Injectable } from '@angular/core';
import { DataStateChangeEventArgs } from '@syncfusion/ej2-angular-treegrid';
import { Subject } from 'rxjs/Subject';
import { Observable } from 'rxjs/Observable';
@Injectable()
export class DataService extends Subject<Object> {
  private BASE_URL = '/api/tasks'; ///// provide the service url required to fetch data from server
  constructor(private http: Http) {
    super();
  }
  public getData(state: DataStateChangeEventArgs): Observable<DataStateChangeEventArgs> {
    const pageQuery = `${skip}=${state.skip}&${stop}=${state.take}`;
    /// filter query for fetching only the root level records
    const treegridQuery = `${filter}='ParentId eq null'`;
    return this.http
      .get(`${this.BASE_URL}?${pageQuery}&${treegridQuery}&${inlinecount}=allpages&${format}=json`)
      .map((response: any) => response.json())
      .map((response: any) => (<DataResult>{
        result: response['d']['results'],
        count: parseInt(response['d']['count'], 10)
      })))
      .map((data: any) => data);
  }
  public execute(state: DataStateChangeEventArgs): void {
    this.getData(state).subscribe(x => {
      super.next(x)
    })
  }
}
```

```
});  
}  
}  
,
```

You can refer to our [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Columns in Angular TreeGrid component

In Syncfusion Angular TreeGrid component, columns are fundamental elements that play a pivotal role in organizing and displaying data within your application. They serve as the building blocks for data presentation, allowing you to specify what data fields to show, how to format and style them, and how to enable various interactions within the tree grid.

The column definitions are used as the [dataSource](#) schema in the tree grid. The [field](#) property of the [column](#) is necessary to map the data source values in tree grid columns.

Column types

The Syncfusion TreeGrid component allows you to specify the type of data that a column binds using the [column.type](#) property. The [type](#) property is used to determine the appropriate [format](#), such as [number](#) or [date](#), for displaying the column data.

Tree Grid supports the following column types:

- **string**: Represents a column that binds to string data. This is the default type if the [type](#) property is not defined.
- **number**: Represents a column that binds to numeric data. It supports formatting options for displaying numbers.
- **boolean**: Represents a column that binds to boolean data. It displays checkboxes for boolean values.
- **date**: Represents a column that binds to date data. It supports formatting options for displaying dates.
- **datetime**: Represents a column that binds to date and time data. It supports formatting options for displaying date and time values.
- **checkbox**: Represents a column that displays checkboxes.

Here is an example of how to specify column types in a tree grid using the types mentioned above.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'  
import { BrowserModule } from '@angular/platform-browser'  
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'  
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'  
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'  
import { Component, OnInit, ViewChild } from '@angular/core';  
import { sampleData } from '../datasource';  
@Component({  
  imports: [  

```

```

        TreeGridModule,
        ButtonModule
    ],
    providers: [PageService,
                SortService,
                FilterService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid #treegrid [dataSource]='data' height='250'
    [treeColumnIndex]='1' childMapping='subtasks' >
        <e-columns>
            <e-column field='taskID' headerText='Task ID'
textAlign='Right' type='number' width=90></e-column>
            <e-column field='taskName' headerText='Task Name'
textAlign='Left' type='string' width=180></e-column>
            <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' type='date' width=120></e-column>
            <e-column field='endDate' headerText='End Date'
textAlign='Right' format='dd/MM/yyyy hh:mm' width=200 type='dateTime'></e-
column>
            <e-column field='approved' headerText='Approved'
width='100' type='boolean' [displayAsCheckBox]='true'></e-column>
            <e-column field='progress' headerText='Progress'
textAlign='Right' type='number' width=120></e-column>
        </e-columns>
    </ejs-treegrid>`
  })
  export class AppComponent implements OnInit {
    public data?: Object[];
    ngOnInit(): void {
      this.data = sampleData;
    }
  }
}

```

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 [type](#) is not defined, then it will be determined from the first record of the [dataSource](#).

* In case if the first record of the [dataSource](#) is null/blank value for a column then it is necessary to define the [type](#) for that column. This is because the tree grid uses the column type to determine which filter dialog to display for that column.

Difference between boolean type and checkbox type column

1. Use the **boolean** column type when you want to bind boolean values from your datasource and/or edit boolean property values from your type.
2. Use the **checkbox** column type when you want to enable selection/deselection of the whole row.

3. When the tree grid column **type** is a **checkbox**, the selection type of the tree grid [selectionSettings](#) will be multiple. This is the default behavior.
4. If you have more than one column with the column type as a **checkbox**, the Tree Grid will automatically enable the other column's checkbox when selecting one column checkbox.

To learn more about how to render boolean values as checkboxes in a Syncfusion TreeGridColumn, please refer to the [Render Boolean Values as Checkbox](#) section.

Column width

To adjust the column width in a Tree Grid, you can use the [width](#) property within the [column](#) property of the Tree Grid configuration. This property enables you to define the column width in pixels or as a percentage. You can set the width to a specific value, like **100** for 100 pixels, or as a percentage value, such as **25%** for 25% of the available width.

1. Tree Grid column width is calculated based on the sum of column widths. For example, a tree grid container with 4 columns and a width of 800 pixels will have columns with a default width of 200 pixels each.
2. If you specify widths for some columns but not others, the Tree Grid will distribute the available width equally among the columns without explicit widths. For example, if you have 3 columns with widths of 100px, 200px, and no width specified for the third column, the third column will occupy the remaining width after accounting for the first two columns.
3. Columns with percentage widths are responsive and adjust their width based on the size of the container. For example, a column with a width of 50% will occupy 50% of the tree grid width and will adjust proportionally when the tree grid container is resized to a smaller size.
4. When you manually resize columns in Syncfusion Tree Grid, a minimum width is set to ensure that the content within the cells remains readable. By default, the minimum width is set to 10 pixels if not explicitly specified for a column.
5. If the total width of all columns exceeds the width of the tree grid container, a horizontal scrollbar will automatically appear to allow horizontal scrolling within the tree grid.
6. When the columns are hidden using the column chooser, the width of the hidden columns is removed from the total tree grid width, and the remaining columns will be resized to fill the available space.
7. If the parent element has a fixed width, the TreeGrid component will inherit that width and occupy the available space. However, if the parent element does not have a fixed width, the TreeGrid component will adjust its width dynamically based on the available space.

To learn more about resizing, you can refer to the resizing section [here](#)

Supported types for column width:

Syncfusion Tree Grid supports the following three types of column width:

1. Auto

The column width is automatically calculated based on the content within the column cells. If the content exceeds the width of the column, it will be truncated with an ellipsis (...) at the end. You can set the width for columns as **auto** in your Tree Grid configuration as shown below:

```
`html
```

```
<e-column field='taskID' headerText='Task ID' textAlign='Right' width='auto'></e-column>
```

2. Percentage

The column width is specified as a percentage value relative to the width of the tree grid container. For example, a column width of 25% will occupy 25% of the total tree grid width. You can set the width for columns as **percentage** in your Tree Grid configuration as shown below:

```
`html
```

```
<e-column field='taskID' headerText='Task ID' textAlign='Right' width='25%'></e-column>
```

3. Pixel

The column width is specified as an absolute pixel value. For example, a column width of 100px will have a fixed width of 100 pixels regardless of the tree grid container size. You can set the width for columns as **pixel** in your Tree Grid configuration as shown below:

```
`html
```

```
<e-column field='taskID' headerText='Task ID' textAlign='Right' width='100'></e-column>
```

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data' height='250'
[treeColumnIndex]='1' childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width='auto'></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width='240'></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=90></e-column>
```

```

        <e-column field='progress' headerText='Progress'
textAlign='Right' width='15%'></e-column>
    </e-columns>
</ejs-treegrid>`
}))
export class AppComponent implements OnInit {
    public data?: Object[];

    ngOnInit(): void {
        this.data = sampleData;
    }
}

```

MAIN.TS

```

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

```

Column formatting

Column formatting is a powerful feature in Syncfusion Tree Grid that allows you to customize the display of data in columns. You can apply different formatting options to columns based on your requirements, such as displaying numbers with specific formats, formatting dates according to a specific locale, and using templates to format column values.

You can use the [column.format](#) property to specify the format for column values.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit, ViewChild } from '@angular/core';
import { formatData } from './datasource';

@Component({
  imports: [

    TreeGridModule,
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' height='250'
[treeColumnIndex]='1' childMapping='subtasks' >
    <e-columns>
        <e-column field='orderId' headerText='Order ID'
textAlign='Right' width=90></e-column>
        <e-column field='orderName' headerText='Order Name'
textAlign='Left' width=180></e-column>
        <e-column field='price' headerText='Price'
textAlign='Right' format='c2' type='number' width=80></e-column>
    </e-columns>
</ejs-treegrid>`
})

```

```

    })
    export class AppComponent implements OnInit {
        public data?: Object[];

        ngOnInit(): void {
            this.data = formatData;
        }
    }
}

```

MAIN.TS

```

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

```

* The Tree Grid uses the [Internalization](#) library to format values based on the specified format and culture.

* By default, the [number](#) and [date](#) values are formatted in **en-US** locale. You can localize the currency and date in different locale as explained [here](#).

* The available format codes may vary depending on the data type of the column.

* You can also customize the formatting further by providing a custom function to the [format](#) property, instead of a format string.

* Make sure that the format string is valid and compatible with the data type of the column, to avoid unexpected results.

Number formatting

Number formatting allows you to customize the display of numeric values in Tree Grid columns. You can use standard numeric format strings or custom numeric format strings to specify the desired format. The [column.format](#) property can be used to specify the number format for numeric columns. For example, you can use the following format strings to format numbers:

Format | Description | Remarks

{ type:'date', format:'dd/MM/yyyy' } | 04/07/1996

{ type:'date', format:'dd.MM.yyyy' } | 04.07.1996

{ type:'date', skeleton:'short' } | 7/4/96

{ type: 'dateTime', format: 'dd/MM/yyyy hh:mm a' } | 04/07/1996 12:00 AM

{ type: 'dateTime', format: 'MM/dd/yyyy hh:mm:ss a' } | 07/04/1996 12:00:00 AM

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit, ViewChild } from '@angular/core';
import { formatData } from './datasource';

@Component({

```



```

imports: [
    TreeGridModule
],
',
standalone: true,
selector: 'app-container',
template: `<ejs-treegrid [dataSource]='data' height='250'
[treeColumnIndex]='1' childMapping='subtasks' >
    <e-columns>
        <e-column field='orderId' headerText='Order ID'
textAlign='Right' width=90></e-column>
        <e-column field='orderName' headerText='Order Name'
textAlign='Left' width=180></e-column>
        <e-column field='orderDate' [format]='formatOptions'
headerText='Order Date' textAlign='Left' width=120></e-column>
        <e-column field='shippedDate' [format]='shipFormat'
headerText='Ship Date' textAlign='Left' width=170></e-column><e-column
field='units' headerText='Units' textAlign='Right' format='N' type='number'
width=80></e-column>
    </e-columns>
</ejs-treegrid>`
))
export class AppComponent implements OnInit {
    public data?: Object[];
    public formatOptions?: object;
    public shipFormat?: object;

    ngOnInit(): void {
        this.data = formatData;
        this.formatOptions = {type: 'date', format: 'dd/MM/yyyy'};
        this.shipFormat = { type: 'dateTime', format: 'dd/MM/yyyy hh:mm a'
    };
}
}

```

MAIN.TS

```

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

```

To learn more about date formatting, you can refer to [Date formatting](#).

Format the date column based on localization

You can also format the date column based on the localization settings of the user's browser. You can use the [format](#) property of the Tree Grid columns along with the [locale](#) property to specify the desired date format based on the locale.

In this example, the format property specifies the date format as "yyyy-MMM-dd", and the locale property specifies the locale as "es-AR" for Spanish (Argentina).

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'

```

```

import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import {
    L10n,
    loadCldr,
    setCulture,
    setCurrencyCode,
} from '@syncfusion/ej2-base';
import { Component, OnInit } from '@angular/core';
import { formatData } from './datasource';
import * as cagregorian from './ca-gregorian.json';
import * as currencies from './currencies.json';
import * as numbers from './numbers.json';
import * as timeZoneNames from './timeZoneNames.json';

@Component({
imports: [

    TreeGridModule,
    ButtonModule

],
providers: [PageService,
            SortService,
            FilterService],
standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid [dataSource]='data' [locale]='locale'
height='315px' [treeColumnIndex]='1' childMapping='subtasks' >
    <e-columns>
        <e-column field='orderId' headerText='Order ID'
textAlign='Right' width=90></e-column>
        <e-column field='orderName' headerText='Order Name'
textAlign='Left' width=180></e-column>
        <e-column field='OrderDate' headerText='OrderDate'
[format]='formatOptions' textAlign='Right' width=150></e-column>
        <e-column field='price' headerText='Price'
textAlign='Right' format='c2' type='number' width=80></e-column>
    </e-columns>
    </ejs-treegrid>`,
})
export class AppComponent implements OnInit {
    public data?: object[];
    public formatOptions?: object;
    public locale: string = 'es-AR';

    ngOnInit(): void {
        setCulture('es-AR');
        setCurrencyCode('ARS');
        loadCldr(cagregorian, currencies, numbers, timeZoneNames);
        this.formatOptions = { type: 'date', format: 'yyyy-MMM-dd' };
        this.data = formatData;
    }
}

```

MAIN.TS

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

Format template column value

Template columns in Tree Grid provide a way to customize the appearance of column values using HTML templates. In addition to HTML markup, you can also use number formatting to format the value displayed in a template column. To format values in a column template, you can use Angular pipes and the [format](#) property. In this example, we are using the date pipe to format the **OrderDate** value as a date in the format **'dd/MMM/yyyy'**.

```
`ts
```

```
<e-column field='orderDate' headerText='Order Date' textAlign='Right' width=120>
```

```
<ng-template #template let-data>
```

```
{% raw %}
```

```
{{ data.orderDate | date:'dd/MMM/yyyy' }}
```

```
{% endraw %}
```

```
</ng-template>
```

```
</e-column>
```

```
,
```

APP.COMPONENT.TS

```
{% raw %}
import { Component, OnInit, ViewChild } from '@angular/core';
import { formatData } from './datasource';
@Component({
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' height='315'
    [treeColumnIndex]='1' childMapping='subtasks' >
    <e-columns>
    <e-column field='orderId' headerText='Order ID' textAlign='Right'
    width=90></e-column>
    <e-column field='orderName' headerText='Order Name' textAlign='Left'
    width=180></e-column>
    <e-column field='orderDate' headerText='Order Date' textAlign='Right'
    width=120>
    <ng-template #template let-data>
    {{ data.orderDate | date:'dd/MMM/yyyy' }}
    </ng-template>
    </e-column>
    <e-column field='shippedDate' headerText='Ship Date' textAlign='Right'
    width=150></e-column>
    <e-column field='units' headerText='Units' textAlign='Right' format='N'
    type='number' width=80></e-column>
    </e-columns>
```

```

</ejs-treegrid>`
  })
  export class AppComponent implements OnInit {
    public data?: Object[];
    ngOnInit(): void {
      this.data = formatData;
    }
  }
  {% enddraw %}

```

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 use other Angular pipes, such as **currency**, **decimal**, **percent**, etc., to format other types of values in the column template based on your requirements.

Custom formatting

Syncfusion Tree Grid allows you to customize the formatting of data in the tree grid columns. You can apply custom formats to numeric or date columns to display data in a specific way according to the requirements. To apply custom formatting to tree grid columns in Syncfusion Tree Grid, you can use the [format](#) property. Here's an example of how you can use custom formatting for numeric and date columns:

In the below example, the **numberFormatOptions** object is used as the **format** property for the **'units'** column to apply a custom numeric format with four decimal places. Similarly, the **dateFormatOptions** object is used as the **format** property for the **'OrderDate'** column to apply a custom date format displaying the date in the format of day-of-the-week, month abbreviation, day, and 2-digit year (e.g. Sun, May 8, '23).

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit, ViewChild } from '@angular/core';
import { formatData } from './datasource';

@Component({
  imports: [
    TreeGridModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' height='250'
    [treeColumnIndex]='1' childMapping='subtasks' >
      <e-columns>
        <e-column field='orderId' headerText='Order ID'
          textAlign='Right' width=90></e-column>

```

```

        <e-column field='orderName' headerText='Order Name'
textAlign='Left' width=180></e-column>
        <e-column field='orderDate' headerText='Order Date'
textAlign='Right' [format]='dateFormatOptions' width=120></e-column>
        <e-column field='units' headerText='Units'
textAlign='Right' [format]='numberFormatOptions' type='number' width=80></e-
column>
    </e-columns>
</ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public numberFormatOptions?: object;
        public dateFormatOptions?: object;
        ngOnInit(): void {
            this.data = formatData;
            this.numberFormatOptions = {
                // Custom format for numeric columns
                format: '##.0000',
            };
            this.dateFormatOptions = {
                // Custom format for date columns
                type: 'Date',
                format: "EEE, MMM d, 'yy",
            };
        }
    }
}

```

MAIN.TS

```

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

```

To learn more about custom formatting, you can refer to [Custom Date formatting](#) and [Custom Number formatting](#).

Align the text of content

You can align the text in the content of a Tree Grid column using the [textAlign](#) property. This property allows you to specify the alignment of the text within the cells of a particular column in the Tree Grid. By default, the text is aligned to the left, but you can change the alignment by setting the value of the [textAlign](#) property to one of the following options:

Left: Aligns the text to the left (default). **Center:** Aligns the text to the center. *Right:* Aligns the text to the right. **Justify:** Align the text to the justify.

Here is an example of using the [textAlign](#) property to align the text of a tree grid column:

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'

```

```

import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, Column } from '@syncfusion/ej2-angular-treegrid';
import { ChangeEventArgs } from '@syncfusion/ej2-angular-dropdowns';
@Component({
  imports: [
    TreeGridModule,
    DropDownListModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div style="display: flex">
    <label style="padding: 30px 17px 0 0;">Select column name
  :</label>
    <ejs-dropdownlist style="padding: 26px 0 0 0" index="0"
width="100" [dataSource]="columnData" [fields]='fields' (change)
="changeColumn($event)"></ejs-dropdownlist>
  </div>

  <div style="display: flex">
    <label style="padding: 30px 17px 0 0;">Align the text for
columns :</label>
    <ejs-dropdownlist style="padding: 26px 0 0 0" index="0"
width="100" [dataSource]="alignmentData" (change)
="changeAlignment($event)"></ejs-dropdownlist>
  </div>
  <ejs-treegrid #treegrid [dataSource]='data' height='250'
[treeColumnIndex]='1' childMapping='subtasks'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
width=80></e-column>
      <e-column field='progress' headerText='Progress'
width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: object[];
  @ViewChild('treegrid') public treegrid?: TreeGridComponent;
  public column_name: any;
  public fields: Object = { text: 'text', value: 'value' };
  public alignmentData: Object[] = [
    { text: 'Left', value: 'Left' },
    { text: 'Right', value: 'Right' },
    { text: 'Center', value: 'Center' },
    { text: 'Justify', value: 'Justify' },
  ];
  public columnData: Object[] = [

```

```

        { text: 'Task Id', value: 'taskID' },
        { text: 'Start Date', value: 'startDate' },
        { text: 'Duration', value: 'duration' },
        { text: 'Progress', value: 'progress' },
    ];
    public changeAlignment(args: ChangeEventArgs): void {
        (this.treegrid as TreeGridComponent).columns.forEach((col: any) => {
            if(col.field == this.column_name){
                col.textAlign = args.value as string;
            }
        });
        (this.treegrid as TreeGridComponent).refreshColumns();
    }
    public changeColumn(args: ChangeEventArgs): void {
        this.column_name=args.value;
    }
    ngOnInit(): void {
        this.data = sampleData;
    }
}

```

MAIN.TS

```

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

```

* The `textAlign` property changes the alignment for both the column content and header. If you want to align header differently, you can use the `headerTextAlign` property.

Render boolean values as checkbox

The TreeGrid component allows you to render boolean values as checkboxes in columns. This can be achieved by using the `displayAsCheckBox` property, which is available in the `column`. This property is useful when you have a boolean column in your Tree Grid and you want to display the values as checkboxes instead of the default text representation of **true** or **false**.

To enable the rendering of boolean values as checkboxes, you need to set the `displayAsCheckBox` property of the `columns` to **true**.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
@Component({
    imports: [

        TreeGridModule,
        ButtonModule
    ]
})

```

```

    ],
    providers: [PageService,
                SortService,
                FilterService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid [dataSource]='data' height='250'
    [treeColumnIndex]='1' childMapping='subtasks'>
        <e-columns>
            <e-column field='taskID' headerText='Task ID'
            textAlign='Right' width=90></e-column>
            <e-column field='taskName' headerText='Task Name'
            textAlign='Left' width=180></e-column>
            <e-column field='startDate' headerText='Start Date'
            textAlign='Right' format='yMd' width=90></e-column>
            <e-column field='approved' headerText='Approved'
            width='150' [displayAsCheckBox]="true" textAlign='Center'> </e-column>
            <e-column field='duration' headerText='Duration'
            textAlign='Right' width=80></e-column>
        </e-columns>
    </ejs-treegrid>`
  })
  export class AppComponent implements OnInit {
    public data?: Object[];
    ngOnInit(): void {
      this.data = sampleData;
    }
  }
}

```

MAIN.TS

```

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

```

* The `displayAsCheckBox` property is only applicable to boolean values in Tree Grid columns.

* When `displayAsCheckBox` is set to **true**, the boolean values will be rendered as checkboxes in the Tree Grid column, with checked state indicating **true** and unchecked state indicating **false**.

How to prevent checkbox in the blank row

To prevent the checkbox in the blank row of the Tree Grid, even if the `displayAsCheckBox` property is set to true for that column, you can use the `rowDataBound` event and check for empty or null values in the row data. If all the values in the row are empty or null, you can set the inner HTML of the corresponding cell to an empty string to hide the checkbox.

Here is an example of how you can prevent a checkbox from being displayed in a blank row in a tree grid:

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'

```



```

import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { RowDataBoundEventArgs, Column } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data' height='315px'
[treeColumnIndex]='1' idMapping='TaskID' parentIdMapping="parentID"
(rowDataBound)='rowDataBound($event)'>
    <e-columns>
      <e-column field='TaskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='TaskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='StartDate' headerText='Start Date'
textAlign='Right' width=160 format='yMd'></e-column>
      <e-column field='Duration' headerText='Duration'
textAlign='Right' format='c2' type='number' width=80></e-column>
      <e-column field='Approved' headerText='Approved'
width=130 displayAsCheckBox="true"></e-column>
    </e-columns>
  </ejs-treegrid>`,
})
export class AppComponent implements OnInit {
  public data?: object[];

  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = projectData;
  }
  rowDataBound(args: RowDataBoundEventArgs) {
    let count = 0;
    let data: any = projectData[(args.data as any).index];
    let keys = Object.keys(data);
    for (let i = 0; i < keys.length; i++) {
      if (
        data[keys[i]] == null ||
        data[keys[i]] == '' ||
        data[keys[i]] == undefined
      ) {
        count++;
      }
    }
  }
}

```

```

    if (count == keys.length || count == keys.length - 1) {
      var col: any = (this.treegrid as TreeGridComponent).columns;
      for (let i = 0; i < col.length; i++) {
        if ((col[i] as Column).displayAsCheckBox) {
          (args.row as Element).children[i].innerHTML = '';
        }
      }
    }
  }
}

```

MAIN.TS

```

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

```

AutoFit columns

The Tree Grid has a feature that allows to automatically adjust column widths based on the maximum content width of each column when you double-click on the resizer symbol located in a specific column header. This feature ensures that all data in the tree grid rows is displayed without wrapping. To use this feature, you need to inject the **ResizeService** in the provider section of **AppModule** and enable the resizer symbol in the column header by setting the [allowResizing](#) property to true in the tree grid.

The following screenshot represents the resizing the column using resizer symbol.

Task ID	Task Name	Start Date	End Date	Duration	Progress	Priority
1	▼ Planning	2/3/2017	2/7/2017	5	100	Normal
2	Plan timeline	2/3/2017	2/7/2017	5	100	Normal
3	Plan budget	2/3/2017	2/7/2017	5	100	Low
4	Allocate reso...	2/3/2017	2/7/2017	5	100	Critical
5	Planning co...	2/7/2017	2/7/2017	0	0	Low
6	▼ Design	2/10/2017	2/14/2017	3	86	High
7	Software Spe...	2/10/2017	2/12/2017	3	60	Normal
8	Develop prot...	2/10/2017	2/12/2017	3	100	Critical
9	Get approval ...	2/13/2017	2/14/2017	2	100	Low

Resizing a column to fit its content using method support

The [autoFitColumns](#) method resizes the column to fit the widest cell's content without wrapping. You can autofit specific columns at initial rendering by invoking the [autoFitColumns](#) method in [dataBound](#) event.

To use [autoFitColumns](#) method, you need to inject **ResizeService** in the provider section of **AppModule**.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, ResizeService } from '@syncfusion/ej2-angular-
treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data' height='250'
[allowResizing]='true' (dataBound)='onDataBound()' [treeColumnIndex]='1'
childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=130></e-column>
      <e-column field='progress' headerText='Progress'
textAlign='Right' width=120></e-column>
    </e-columns>
  </ejs-treegrid>`,
  providers: [ResizeService ]
})
export class AppComponent implements OnInit {
  public data?: Object[];
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
  }
  onDataBound() {
    (this.treeGridObj as
TreeGridComponent).autoFitColumns(['taskName']);
  }
}

```

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 autofit all the columns by invoking the `autoFitColumns` method without specifying column names.

AutoFit columns with empty space

The Autofit feature is utilized to display columns in a tree grid based on the defined width specified in the columns declaration. If the total width of the columns is less than the width of the tree grid, this means that white space will be displayed in the tree grid instead of the columns auto-adjusting to fill the entire tree grid width.

You can enable this feature by enabling the `autoFit` property of grid object in tree grid instance using the `load` event of the tree grid. This feature ensures that the column width is rendered only as defined in the tree grid column definition.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit, ViewChild } from '@angular/core';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { formatData } from './datasource';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data' height='250'
[treeColumnIndex]='1' childMapping='subtasks' allowResizing= 'true' width=
'850' (load) ="load()">
    <e-columns>
      <e-column field='orderId' headerText='Order ID'
textAlign='Right' minWidth='100' width='150' maxWidth='200'></e-column>
      <e-column field='orderName' headerText='Order Name'
textAlign='Left' minWidth='100' width='150' maxWidth='200'></e-column>
      <e-column field='orderDate' headerText='Order Date'
textAlign='Right' minWidth='100' width='150' maxWidth='200'
format='yMd'></e-column>
      <e-column field='price' headerText='Price'
textAlign='Right' format='c2' type='number' minWidth='100' width='150'
maxWidth='200'></e-column>
    </e-columns>
  </ejs-treegrid>`,
})
```

```
export class AppComponent implements OnInit {
  public data?: Object[];
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  ngOnInit(): void {
    this.data = formatData;
  }
  load(): void {
    (this.treeGridObj as TreeGridComponent).grid.autoFit = true;
  }
}
```

MAIN.TS

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

If any one of the column width is undefined, then the particular column will automatically adjust to fill the entire width of the tree grid table, even if you have enabled the `autoFit` property of tree grid.

AutoFit columns when changing column visibility using column chooser

In Syncfusion Tree Grid, you can auto-fit columns when the column visibility is changed using the column chooser. This can be achieved by calling the `autoFitColumns` method in the `actionComplete` event. By using the `requestType` property in the event arguments, you can differentiate between different actions, and then call the `autoFitColumns` method when the request type is `columnState`.

Here's an example code snippet in Angular that demonstrates how to auto fit columns when changing column visibility using column chooser:

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService,
  FilterService, ColumnChooserService, ToolbarService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, TreeActionEventArgs, } from '@syncfusion/ej2-
angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService, ColumnChooserService, ToolbarService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
```

```

    template: `<ejs-treegrid #treegrid [dataSource]='data' height='250'
    [treeColumnIndex]='1' childMapping='subtasks' [allowPaging]="true"
    [toolbar]="toolbarItems"
    (actionComplete)="onActionComplete($event)" [showColumnChooser]= 'true' >
        <e-columns>
            <e-column field='taskID' headerText='Task ID'
            textAlign='Right' width=90></e-column>
            <e-column field='taskName' headerText='Task Name'
            textAlign='Left' width=180></e-column>
            <e-column field='startDate' headerText='Start Date'
            textAlign='Right' width=160 format='yMd'></e-column>
            <e-column field='duration' headerText='Duration'
            textAlign='Right' format='c2' type='number' width=80></e-column>
        </e-columns>
    </ejs-treegrid>`
  })
  export class AppComponent implements OnInit {
    public data?: object[];
    @ViewChild('treegrid')
    public treegrid?: TreeGridComponent;
    public toolbarItems?: string[];
    ngOnInit(): void {
      this.data = sampleData;
      this.toolbarItems = ['ColumnChooser'];
    }
    public onActionComplete({ requestType }: TreeActionEventArgs): void {
      if (requestType === 'columnstate') {
        (this.treegrid as TreeGridComponent).autoFitColumns();
      }
    }
  }
}

```

MAIN.TS

```

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

```

Locked columns

The Syncfusion Tree Grid allows you to lock columns, which prevents them from being reordered and moves them to the first position. This functionality can be achieved by setting the [column.lockColumn](#) property to **true**, which locks the column and moves it to the first position in the tree grid.

Here's an example of how you can use the `lockColumn` property to lock a column in the Syncfusion tree grid. Additionally, in the code snippet below, differentiate the locked column using CSS with a custom attributes property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridModule, ReorderService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewEncapsulation } from '@angular/core';

```

```

import { sampleData } from './datasource';
import { ReorderService } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule
  ],
  providers: [ReorderService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data' height='285'
[allowReordering]='true' [allowSelection]='false' [treeColumnIndex]='2'
childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=120 [lockColumn]='true'
[customAttributes]='customAttributes'></e-column>
      <e-column field='progress' headerText='Progress'
textAlign='Right' width=120></e-column>
    </e-columns>
  </ejs-treegrid>`,
  providers: [ReorderService],
  encapsulation: ViewEncapsulation.None,
  styleUrls: ['./custom-column.style.css']
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public customAttributes?: Object;
  ngOnInit(): void {
    this.data = sampleData;
    this.customAttributes = {class: 'customcss'};
  }
}

```

MAIN.TS

```

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

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion Angular Grid</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Typescript UI Controls" />

```

```

<meta name="author" content="Syncfusion" />
<style>
  #loader {
    color: #008cff;
    font-family: 'Helvetica Neue', 'calibiri';
    font-size: 16px;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
  }
</style>
</head>
<body style="margin-top: 125px">
  <app-container>
    <div id='loader'>Loading....</div>
  </app-container>
</body>
</html>

```

Show or hide columns

The Syncfusion TreeGrid component allows you to show or hide columns dynamically by using property or methods available in the tree grid. This feature can be useful when you want to customize the visibility of columns in the Tree Grid based on the requirements.

Using property

You can show or hide columns in the Tree Grid using the [visible](#) property of each column. By setting the **visible** property to **true** or **false**, you can control whether the column should be visible or hidden in the tree grid. Here's an example of how to show or hide a column in the tree grid using the visible property:

In the below example, the **duration** column is defined with **visible** property set to **false**, which will hide the column in the rendered tree grid.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule, ReorderService } from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
@Component({
  imports: [

    TreeGridModule
  ],
  providers: [ReorderService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' height='250'
[treeColumnIndex]='1' childMapping='subtasks' autoCheckHierarchy='true'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>

```



```

        <e-column field='taskName' headerText='Task Name'
        textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
        textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
        [visible]='false' textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: object[];
        ngOnInit(): void {
            this.data = sampleData;
        }
    }
}

```

MAIN.TS

```

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

```

* Hiding a column using the `visible` property only affects the UI representation of the tree grid. The data for the hidden column will still be available in the underlying data source, and can be accessed or modified programmatically.

* When a column is hidden, its width is not included in the calculation of the total tree grid width.

* To hide a column permanently, you can set its visible property to false in the column definition, or remove the column definition altogether.

Using methods

You can also show or hide columns in tree grid using the [showColumns](#) and [hideColumns](#) methods of the TreeGrid component. These methods allow you to show or hide columns based on either the `headerText` or the `field` of the column.

Based on header text

You can dynamically show or hide columns in the Tree Grid based on the header text by invoking the `showColumns` or `hideColumns` methods. These methods take either an array of column header texts or a simple string value for a column header text as the first parameter, and the value `headerText` as the second parameter to specify that you are showing or hiding columns based on the header text.

Here's an example of how to show or hide a column based on the HeaderText in the tree grid:

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridModule, ReorderService } from '@syncfusion/ej2-angular-treegrid';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';

```

```

import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [

    TreeGridModule, ButtonModule
  ],
  providers: [ReorderService],
  standalone: true,
  selector: 'app-container',
  template: `<button id='show' ej2-button class='e-flat' (click)='show()'>
Show </button>
    <button id='hide' ej2-button class='e-flat' (click)='hide()'>
Hide </button>
    <ejs-treegrid #treegrid [dataSource]='data' height='285'
[treeColumnIndex]='1' childMapping='subtasks' >
    <e-columns>
    <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
    <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
    <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
    <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
    </ejs-treegrid>`,
  })
export class AppComponent implements OnInit {
  public data?: Object[];
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;

  ngOnInit(): void {
    this.data = sampleData;
  }
  show() {
    (this.treeGridObj as TreeGridComponent).showColumns(
      'Task Name',
      'headerText'
    ); //show by HeaderText
  }
  hide() {
    (this.treeGridObj as TreeGridComponent).hideColumns(
      'Task Name',
      'headerText'
    ); //hide by HeaderText
  }
}

```

MAIN.TS

```

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

```

Based on field

You can dynamically show or hide columns in the Tree Grid using external buttons based on the field by invoking the `showColumns` or `hideColumns` methods. These methods take either an array of column header texts or a simple string value for a column header text as the first parameter, and the value `field` as the second parameter to specify that you are showing or hiding columns based on the field.

Here's an example of how to show or hide a column based on the field in the tree grid:

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule, ReorderService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [
    ButtonModule,
    TreeGridModule
  ],
  providers: [ReorderService],
  standalone: true,
  selector: 'app-container',
  template: `<button id='show' ej-button class='e-flat' (click)='show()'>
Show </button>
<button id='hide' ej-button class='e-flat' (click)='hide()'>
Hide </button>
<ejs-treegrid #treegrid [dataSource]='data' height='285'
[treeColumnIndex]='1' childMapping='subtasks' >
<e-columns>
<e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
<e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
<e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
<e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
</e-columns>
</ejs-treegrid>`,
})
export class AppComponent implements OnInit {
  public data?: Object[];

  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
  }
  show() {
```

```

    (this.treeGridObj as TreeGridComponent).showColumns(['taskName',
'startDate'], 'field'); //show by Field
    }
    hide() {
        (this.treeGridObj as TreeGridComponent).hideColumns(['taskName',
'startDate'], 'field'); //hide by Field
    }
}

```

MAIN.TS

```

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

```

Controlling Tree Grid actions

You can control various actions such as filtering, sorting, resizing, reordering, editing, and searching for specific columns in the Syncfusion Angular Tree Grid using the following properties:

- [allowEditing](#): Enables or disables editing for a column.
- [allowFiltering](#): Enables or disables filtering for a column.
- [allowSorting](#): Enables or disables sorting for a column.
- [allowReordering](#): Enables or disables reordering for a column.
- [allowResizing](#): Enables or disables resizing for a column.
- [allowSearching](#): Enables or disables searching for a column.

Here is an example code that demonstrates how to control tree grid actions for specific columns:

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data' height='250'
[treeColumnIndex]='1' childMapping='subtasks' [allowFiltering]="true"
[allowSorting]="true">
    <e-columns>

```

```

        <e-column field='taskID' headerText='Task ID'
[allowSorting]="false" textAlign='Right' width=120></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
[allowFiltering]="false" textAlign='Right' format='yMd' width=120></e-
column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=120></e-column>
        <e-column field='progress' headerText='Progress'
textAlign='Right' width=120></e-column>
        </e-columns>
    </ejs-treegrid>`,
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        ngOnInit(): void {
            this.data = sampleData;
        }
    }
}

```

MAIN.TS

```

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

```

Customize column styles

Customizing the tree grid column styles allows you to modify the appearance of columns in the TreeGrid component to meet your design requirements. You can customize the font, background color, and other styles of the columns. To customize the columns styles in the tree grid, you can use tree grid event, CSS, property or method support.

For more information check on this [documentation](#).

Manipulating columns

The Syncfusion Tree Grid for Angular provides powerful features for manipulating columns. This section explains how to access columns, update column definitions, and add/remove columns using Syncfusion Tree Grid properties, methods, and events.

Accessing columns

To access columns in the Syncfusion TreeGrid component, you can use the following methods:

- [getColumns](#):

This method returns the array of columns defined in the Tree Grid.

```
`ts
```

```
let columns = this.treegrid.getColumns();
```

```
,
```

- [getColumnByField:](#)

This method returns the column object that matches the specified field name.

```
`ts
```

```
let column = this.treegrid.getColumnByField('taskName');
```

```
,
```

- [getColumnByUid:](#)

This method returns the column object that matches the specified UID.

```
`ts
```

```
let column = this.treegrid.getColumnByUid("grid-column128");
```

```
,
```

- [getVisibleColumns:](#)

This method returns the array of visible columns.

```
`ts
```

```
let visibleColumns = this.treegrid.getVisibleColumns();
```

```
,
```

- [getColumnFieldNames](#)

This method returns an array of field names of all the columns in the Tree Grid.

```
`ts
```

```
let fieldNames = this.treegrid.getColumnFieldNames()
```

```
,
```

For a complete list of column properties, refer to this [section](#).

Updating column definitions

You can update the column definitions in the Tree Grid using the [column](#) property. You can modify the properties of the column objects in the columns array to update the columns dynamically. For example, you can change the headerText, width, visible, and other properties of a column to update its appearance and behavior in the tree grid and then call the [refreshColumns](#) method to apply the changes to the tree grid.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
```

```

import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<button ej-button id="btnId" cssClass="e-info"
(click)="updateColumns()"> Update Columns </button>
    <ejs-treegrid #treegrid [dataSource]='data' height='250'
[treeColumnIndex]='1' childMapping='subtasks' >
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=130></e-column>
        <e-column field='progress' headerText='Progress'
textAlign='Right' width=120></e-column>
      </e-columns>
    </ejs-treegrid>`,
})
export class AppComponent implements OnInit {
  public data?: Object[];
  @ViewChild('treegrid') treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
  }
  updateColumns(): void {
    // Modifying column properties
    var column: any = (this.treegrid as TreeGridComponent).columns;
    column[0].textAlign = 'Center';
    column[0].width = '100';
    column[2].visible = false;
    column[1].customAttributes = {
      class: 'customcss',
    };
    // Applying changes to the treegrid
    (this.treegrid as TreeGridComponent).refreshColumns();
  }
}

```

MAIN.TS

```

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

```

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

Adding/removing columns

The TreeGrid component allows you to dynamically add or remove columns to and from the tree grid using the [columns](#) property, which can be accessed through the instance of the Tree Grid.

To add a new column to the Tree Grid, you can directly **push** the new column object to the columns property. To remove a column from the Tree Grid, you can use the **pop** method, which removes the last element from the columns array of the Tree Grid. Alternatively, you can use the splice method to remove a specific column from the columns array.

Here's an example of how you can add and remove a column from the tree grid:

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, Column } from '@syncfusion/ej2-angular-
treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<button ej-button id='add' cssClass="e-info"
(click)='addColumns()'> Add Column</button>
    <button ej-button id='delete' cssClass="e-info"
(click)='deleteColumns()'> Delete Column</button>

    <ejs-treegrid #treegrid [dataSource]='data' height='250'
[treeColumnIndex]='1' childMapping='subtasks'>
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
      </e-columns>
    </ejs-treegrid>`,
```



```

})
export class AppComponent implements OnInit {
  public data?: object[];
  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;

  ngOnInit(): void {
    this.data = sampleData;
  }
  addColumns(): void {
    var col: any = (this.treegrid as TreeGridComponent).columns;
    var newColumns = [
      { field: 'progress', headerText: 'Progress', width: 120 },
      { field: 'priority', headerText: 'Priority', width: 120, format: 'yMd'
    },
    ];
    newColumns.forEach((cols: any) => {
      col.push(cols);
    });
    (this.treegrid as TreeGridComponent).refreshColumns();
  }
  deleteColumns(): void {
    var col: any = (this.treegrid as TreeGridComponent).columns;
    col.pop();
    (this.treegrid as TreeGridComponent).refreshColumns();
  }
}

```

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 refresh columns

You can use the [refreshColumns](#) method of the Syncfusion Tree Grid to refresh the columns in the tree grid. This method can be used when you need to update the tree grid columns dynamically based on user actions or data changes.

```
`ts
```

```
this.treegrid.refreshColumns();
```

```
,
```

Responsive columns

The Syncfusion Angular TreeGrid component provides a built-in feature to toggle the visibility of columns based on media queries using the [hideAtMedia](#) property of the column object. The `hideAtMedia` accepts valid [Media Queries](#).

In this example, we have a tree grid that displays data with two columns: **Task ID and Start Date**. We have set the `hideAtMedia` property of the **Task ID** column to (min-width: 700px) which means that this column will be hidden when the browser screen width is less than or equal to 700px.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
              SortService,
              FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' height='250'
[treeColumnIndex]='1' childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
hideAtMedia='(min-width: 700px)' textAlign='Right' width=90></e-column> //
column hides when browser screen width lessthan 700px;
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
hideAtMedia='(max-width: 500px)' textAlign='Right' format='yMd'
width=90></e-column> // column shows when browser screen width lessthan or
equalto 500px;
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  ngOnInit(): void {
    this.data = sampleData;
  }
}

```

MAIN.TS

```

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

```

[Row in Angular TreeGrid component](#)

Each row typically represents a single record or item from a data source. Rows in a tree grid are used to present data in a tabular format. Each row displays a set of values representing the fields of an

individual data record. Rows allow to interact with the data in the tree grid. You can select rows, edit cell values, perform sorting or filtering operations, and trigger events based on actions.

Customize row styles

Customizing the styles of rows in a Syncfusion Tree Grid allows you to modify the appearance of rows to meet your design requirements. This feature is useful when you want to highlight certain rows or change the font style, background color, and other properties of the row to enhance the visual appeal of the tree grid. To customize the row styles in the tree grid, you can use CSS, properties, methods, or event support provided by the Syncfusion Angular TreeGrid component.

Using event

You can customize the appearance of the rows by using the [rowDataBound](#) event. This event triggers for every row when it is bound to the data source. In the event handler, you can get the [RowDataBoundEventArgs](#) object, which contains details of the row. You can use this object to modify the row's appearance, add custom elements, or perform any other customization.

Here's an example illustrating how you can utilize the `rowDataBound` event to customize the styles of rows based on the value of the **duration** column. This example involves inspecting the value of the Duration column for each row and adjusting a row's style accordingly.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
import { RowDataBoundEventArgs } from '@syncfusion/ej2-grids';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='250' [enableHover]='false' (rowDataBound)='rowBound($event)'
childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  `
})
export class AppComponent {
  rowBound(event: RowDataBoundEventArgs) {
    // Custom logic to modify row style based on duration
  }
}
```

```

        </e-columns>
    </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        ngOnInit(): void {
            this.data = sampleData;
        }
        rowBound(args: RowDataBoundEventArgs | any) {
            if (args.data['duration'] == 0 ) {
                args.row.style.background= '#336c12';
            } else if (args.data['duration'] < 3) {
                args.row.style.background= '#7b2b1d';
            }
        }
    }
}

```

MAIN.TS

```

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

```

The [queryCellInfo](#) event can also be used to customize cells and is triggered for every cell in the tree grid. It can be useful when you need to customize cells based on certain conditions or criteria.

Using CSS

You can apply styles to the rows using CSS selectors. The Tree Grid provides a class name for each row element, which you can use to apply styles to that specific row.

Customize selected row

You can customize the appearance of the selected row using CSS. This is useful when you want to highlight the currently selected row for improve the visual appeal of the Tree Grid. By default, the Tree Grid provides the CSS class **.e-selectionbackground** to style the selected row. You can customize this default style by overriding the **.e-selectionbackground** class with your own custom CSS styles.

To change the background color of the selected row, you can add the following CSS code to your application:

```

`css
.e-treegrid .e-selectionbackground {
background-color: #f9920b;
}
`

```

Here's an example of how to use the **.e-selectionbackground** class to style the selected row:

APP.COMPONENT.TS

```

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

```

```

import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import { sampleData } from './datasource';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  encapsulation: ViewEncapsulation.None,
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='250' [enableHover]='false' childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`,
  styles: [
    `.e-treegrid .e-selectionbackground {
      background-color: #f9920b !important;
    }`,
  ],
})
export class AppComponent implements OnInit {
  public data?: Object[];
  ngOnInit(): void {
    this.data = sampleData;
  }
}

```

MAIN.TS

```

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

```

Using method

The Tree Grid provides below methods to customize the appearance of the tree grid rows :

1. [getRowByIndex](#): This method returns the HTML element of a row at the specified index. You can use this method to apply custom styles to a specific row.
2. [getRows](#): This method returns an array of all the row elements in the Tree Grid. You can use this method to apply custom styles to all rows or to a specific set of rows based on some condition.
3. [getRowInfo](#): This method returns the data object and index of the row corresponding to the specified row element. You can use this method to apply custom styles based on the data in a row.
4. [getSelectedRowIndexes](#): This method returns an array of the indexes of the selected rows in the Tree Grid. You can use this method to apply custom styles to the selected rows.
5. [getSelectedRows](#): This method returns an array of the HTML elements representing the selected rows in the tree grid. You can use this method to directly loop through the selected rows and customize their styles.

The following example demonstrates how to use `getRowByIndex` method to customize the appearance of the row inside the `dataBound` event of the tree grid.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild, ViewEncapsulation } from '@angular/core';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { sampleData } from './datasource';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  encapsulation: ViewEncapsulation.None,
  template: `<ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' height='250' [enableHover]='false'
childMapping='subtasks' (dataBound)="customizeRows()">
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  `
})
export class AppComponent {
  customizeRows() {
    // Customization logic here
  }
}
```

```

        </e-columns>
    </ejs-treegrid>`,
  })
  export class AppComponent implements OnInit {
    public data?: Object[];
    @ViewChild('treegrid')
    public treegrid: TreeGridComponent | undefined;
    ngOnInit(): void {
      this.data = sampleData;
    }
    public customizeRows() {
      (
        (this.treegrid as TreeGridComponent).getRowByIndex(2) as HTMLElement
      ).style.background = 'rgb(193, 228, 234)';
    }
  }
}

```

MAIN.TS

```

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

```

Styling alternate rows

You can customize the appearance of the alternate rows using CSS. This can be useful for improving the readability of the data and making it easier to distinguish between rows. By default, Syncfusion Tree Grid provides the CSS class **.e-altrow** to style the alternate rows. You can customize this default style by overriding the **.e-altrow** class with your custom CSS styles.

To change the background color of the alternate rows, you can add the following CSS code to your application's stylesheet:

```

`css

.e-treegrid .e-altrow {
background-color: #fafafa;
}
`

```

Please refer to the following example.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import { sampleData } from './datasource';
@Component({

```

```

imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
],
providers: [PageService,
    SortService,
    FilterService],
standalone: true,
selector: 'app-container',
encapsulation: ViewEncapsulation.None,
template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='250' [enableHover]='false' childMapping='subtasks' >
    <e-columns>
        <e-column field='taskId' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
    </ejs-treegrid>`,
    styles: [
        `.e-treegrid .e-altrow {
            background-color: #fafafa;
        }`,
    ],
})
export class AppComponent implements OnInit {
    public data?: Object[];
    ngOnInit(): void {
        this.data = sampleData;
    }
}

```

MAIN.TS

```

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

```

Styling parent and child rows

You can customize the appearance of the parent and child rows by utilizing the [rowDataBound](#) event. This event is triggered for every row when it is bound to the data source. Within the event handler, you can access the [RowDataBoundEventArgs](#) object, which contains details of the row. Use this object to modify the parent and child row's appearance, add custom elements, or perform any other customization by using condition.

Here's an example illustrating how to utilize the `rowDataBound` event to customize the styles of parent and child rows based on the `hasChildRecords` property value in the `dataSource` object of the tree grid. In this example, the style of each row is adjusted based on the **hasChildRecords** property.

Please refer to the following example.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
import { RowDataBoundEventArgs } from '@syncfusion/ej2-grids';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='250' [enableHover]='false' (rowDataBound)='rowBound($event)'
childMapping='subtasks' >
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`,
})
export class AppComponent implements OnInit {
  public data?: Object[];
  ngOnInit(): void {
    this.data = sampleData;
  }
  rowBound(args: RowDataBoundEventArgs | any) {
    if (!args.data.hasChildRecords) {
      //Here apply the background color of child rows
      args.row.style.background = '#33ff12';
    } else {
      //Here apply the background color of parent rows
      args.row.style.background = '#ff2f1f';
    }
  }
}
```

```

    }
  }
}

```

MAIN.TS

```

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

```

Row height

The Syncfusion Tree Grid allows you to customize the height of rows based on your needs. This feature can be useful when you need to display more content in a row or when you want to reduce the height of rows to fit its content. You can achieve this by using the [rowHeight](#) property of the TreeGrid component. This property allows you to change the height of the entire tree grid row to your desired value.

In the following example, the demonstration illustrates how to dynamically alter the row height using the `rowHeight` property.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild, ViewEncapsulation } from
'@angular/core';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { sampleData } from './datasource';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<div>
    <button ej-button id="small" cssClass="e-small"
(click)="clickHandler($event)">
      Change height 20px</button>
    <button ej-button id="medium" cssClass="e-small"
(click)="clickHandler($event)">
      Default height 42px</button>
    <button ej-button id="big" cssClass="e-small"
(click)="clickHandler($event)">
      Change height 60px</button>

```

```

        </div>
        <div class="control-section" style="padding-top:20px">
            <ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' height='250' [enableHover]='false' rowHeight='42'
childMapping='subtasks' >
                <e-columns>
                    <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
                    <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
                    <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
                    <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
                </e-columns>
            </ejs-treegrid>
        </div>`,
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        @ViewChild('treegrid')
        public treegrid: TreeGridComponent | undefined;
        public heightRow: { [key: string]: number } = {
            small: 20,
            medium: 40,
            big: 60,
        };
        ngOnInit(): void {
            this.data = sampleData;
        }
        clickHandler(args: MouseEvent): void {
            (this.treegrid as TreeGridComponent).rowHeight =
                this.heightRow[(args.target as HTMLElement).id];
        }
    }
}

```

MAIN.TS

```

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

```

- * The **rowHeight** property can only be used to set the height of the entire tree grid row. It cannot be used to set the height of individual cells within a row.
- * The **rowHeight** property applies the height to all rows in the tree grid, including the header and footer rows.
- * You can also set the height for a specific row using the **rowHeight** property of the corresponding row object in the **rowDataBound** event.

Customize row height for particular row

Customizing the row height for a particular row can be useful when you want to display more content in a particular row, reduce the height of a row to fit its content, or make a specific row stand out from the

other rows in the tree grid. This can be achieved by using the [rowHeight](#) property of the TreeGrid component along with the [rowDataBound](#) event.

The `rowHeight` property of the TreeGrid component allows you to set the height of all rows in the tree grid to a specific value. However, if you want to customize the row height for a specific row based on the row data, you can use the `rowDataBound` event. This event is triggered every time a request is made to access row information, element, or data, and before the row element is appended to the Tree Grid element.

In the below example, the row height for the row with `taskID` as '5' is set as '90px' using the `rowDataBound` event.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild, ViewEncapsulation } from
'@angular/core';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { RowDataBoundEventArgs } from '@syncfusion/ej2-grids';
import { sampleData } from '../datasource';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' height='250' childMapping='subtasks'
(rowDataBound)='rowDataBound($event)' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`,
})
export class AppComponent implements OnInit {
  public data?: Object[];
  @ViewChild('treegrid')
```

```

public treegrid: TreeGridComponent | undefined;
ngOnInit(): void {
    this.data = sampleData;
}
public rowDataBound(args: RowDataBoundEventArgs) {
    if ((args.data as any)['taskID'] === 5) {
        args.rowHeight = 90;
    }
}
}

```

MAIN.TS

```

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

```

* In virtual scrolling mode, it is not applicable to set different row heights.

* You can customize the row height of multiple rows by checking the relevant criteria in the `rowDataBound` event and setting the `rowHeight` property accordingly.

* In the `rowDataBound` event handler, you can access the current row using the `args.row` property and set the `rowHeight` property for that row using the `setAttribute` method.

Row hover

The Row Hover feature in Tree Grid provides a visual effect when the mouse pointer hovers over the rows, making it easy to highlight and identify the selected row. This feature can also improve the readability of data in the tree grid. The row hover effect can be enabled or disabled using the `enableHover` property of the TreeGrid component.

By default, the `enableHover` property is set to **true**, which means that the row hovering effect is enabled. To disable the row hover effect, set the `enableHover` property to **false**.

Here is an example that demonstrates how to enable/disable the Row Hover feature:

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { SwitchModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit, ViewChild, ViewEncapsulation } from '@angular/core';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { sampleData } from './datasource';
@Component({
    imports: [

        TreeGridModule,

```

```

        ButtonModule,
        DropDownListAllModule, SwitchModule
    ],
    providers: [PageService,
                SortService,
                FilterService],
    standalone: true,
    selector: 'app-container',
    template: `<div style="padding:0px 0px 20px 0px">

        <label> Enable/Disable Row Hover</label>
        <ejs-switch id="switch" [(checked)]="enableRowHover"
        (change)="toggleRowHover()"></ejs-switch>
    </div>
    <ejs-treegrid #treegrid [dataSource]='data'
    [treeColumnIndex]='1' height='250' childMapping='subtasks'
    [enableHover]="enableRowHover" >
        <e-columns>
            <e-column field='taskID' headerText='Task ID'
            textAlign='Right' width=90></e-column>
            <e-column field='taskName' headerText='Task Name'
            textAlign='Left' width=180></e-column>
            <e-column field='startDate' headerText='Start Date'
            textAlign='Right' format='yMd' width=90></e-column>
            <e-column field='duration' headerText='Duration'
            textAlign='Right' width=80></e-column>
        </e-columns>
    </ejs-treegrid>`,
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        @ViewChild('treegrid')
        public treegrid: TreeGridComponent | undefined;
        public enableRowHover: boolean = true;
        ngOnInit(): void {
            this.data = sampleData;
        }
        toggleRowHover(): void {
            this.enableRowHover = !this.enableRowHover;
        }
    }
}

```

MAIN.TS

```

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

```

The `enableHover` property applies to the entire tree grid, not individual rows or columns.

Custom actions or items on row hover

You can execute custom actions or display custom items when hovering over rows by using the [rowDataBound](#) event of the tree grid.

The `rowDataBound` event is triggered every time a request is made to access row information, element, or data, before the row element is appended to the Tree Grid element.

Here's an illustrative example demonstrating how to implement a custom action using the `rowDataBound` event. In this event, when hovering over a row, a tooltip with a button is displayed. Clicking the button reveals a custom message.

APP.COMPONENT.TS

```
{% raw %}
import { Component, OnInit, ViewChild, ViewEncapsulation } from
 '@angular/core';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { RowDataBoundEventArgs } from '@syncfusion/ej2-grids';
import { sampleData } from './datasource';
import { Tooltip } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
@Component({
  selector: 'app-container',
  template: `<div id='show' style="padding:0px 0px 20px 0px;" >
</div>
<p id="message">{{message}}</p>
<ejs-treegrid #treegrid [dataSource]='data' [treeColumnIndex]='1'
height='315' childMapping='subtasks' (rowDataBound)='rowDataBound($event)' >
<e-columns>
<e-column field='taskID' headerText='Task ID' textAlign='Right'
width=90></e-column>
<e-column field='taskName' headerText='Task Name' textAlign='Left'
width=180></e-column>
<e-column field='startDate' headerText='Start Date' textAlign='Right'
format='yMd' width=90></e-column>
<e-column field='duration' headerText='Duration' textAlign='Right'
width=80></e-column>
</e-columns>
</ejs-treegrid>`,
})
export class AppComponent implements OnInit {
  public data?: Object[];
  @ViewChild('treegrid')
  public treegrid: TreeGridComponent | undefined;
  public message: string = '';
  ngOnInit(): void {
    this.data = sampleData;
  }
  rowDataBound(args: RowDataBoundEventArgs): void {
    //Here bind mouse over event while hovering over the row
    (args.row as HTMLElement).addEventListener(
      'mouseover',
      (mouseargs: MouseEvent) => {
        //Here button creation
        const buttonId = 'element_' + (args.data as any)['taskID'];
        const buttonElement = document.createElement('button');
        buttonElement.id = buttonId;
        buttonElement.textContent = 'Row details';
        let target: any;
        //Set tooltip target
        if (mouseargs && mouseargs.target !== null) {
```

```

if ((mouseargs.target as any).classList.contains('e-rowcell')) {
  target = mouseargs.target;
} else {
  target = null;
}
}
//Tooltip creation
const tooltip: Tooltip = new Tooltip(
{
  content: buttonElement,
  width: '100px',
  height: '40px',
  opensOn: 'Hover',
},
target
);
//Button click event
buttonElement.addEventListener('click', () => {
  // Handle button click here
  this.message = `Button clicked for task ID: ${
    (args.data as any)['taskID']
  }`;
});
}
);
}
}
{% enddraw %}

```

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 get the row information when hovering over the cell

You can retrieve row information when hovering over a specific cell. This can be useful if you want to display additional details or perform some action based on the data in the row. This can be achieved by using the [rowDataBound](#) event and the [getRowInfo](#) method of the Tree Grid.

- The `rowDataBound` event is triggered every time a request is made to access row information, element, or data, before the row element is appended to the Tree Grid element.
- The `getRowInfo` method is used to retrieve the row information when hovering over a specific cell. This method takes a single parameter, which is the target element that is being hovered over.

Here's an example that demonstrates how to use the `rowDataBound` event and `getRowInfo` method to retrieve the row information when hovering over a cell in the Syncfusion Tree Grid.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'

```



```

import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild, ViewEncapsulation } from
'@angular/core';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { RowDataBoundEventArgs } from '@syncfusion/ej2-grids';
import { sampleData } from './datasource';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<div id='show' style="padding:0px 0px 20px 0px;" >
    </div>
    <ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' height='250' childMapping='subtasks'
(rowDataBound)='rowDataBound($event)' >
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
      </e-columns>
    </ejs-treegrid>`,
})
export class AppComponent implements OnInit {
  public data?: Object[];
  @ViewChild('treegrid')
  public treegrid: TreeGridComponent | undefined;
  ngOnInit(): void {
    this.data = sampleData;
  }
  rowDataBound(args: RowDataBoundEventArgs): void {
    (args.row as HTMLElement).addEventListener('mouseover', (e: MouseEvent)
=> {
      const rowInformation = (this.treegrid as
TreeGridComponent).getRowInfo(
        e.target as HTMLElement
      );
      console.log(rowInformation);
      (document.getElementById('show') as HTMLElement).innerHTML = `
        <table style="border-collapse: collapse; width: 600px;">

```

```

        <tr style="border: 2px solid;">
            <td style="padding: 2px;"><b>Row Information:</b></td>
        </tr>
        <tr style="border: 2px solid; padding: 8px;">
            <th style="border: 2px solid; padding: 8px; width:
120px;"><b>Class Name</b>
            </th>
            <td style="border: 2px solid; padding: 8px;">${
                (rowInformation.row as Element).className
            }
            </td>
        </tr>
        <tr style="border: 2px solid;">
            <th style="border: 2px solid; padding: 8px;"><b>Row Index</b>
            </th>
            <td style="border: 2px solid; padding: 8px;">${
                rowInformation.rowIndex
            }
            </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 `getRowInfo` method can only be used in the `rowDataBound` event. Attempting to use it elsewhere will result in an error.

Adding a new row programmatically

The Syncfusion Tree Grid provides a way to add a new row to the tree grid programmatically. This feature is useful when you want to add a new record to the tree grid without having the manually enter data in the tree grid. This can be done using the [addRecord](#) method of the Tree Grid.

The `addRecord` method takes three parameters:

- The **data** object representing the new row to be added
- The **index** at which the new row should be inserted. If no index is specified, the new row will be added at the end of the Tree Grid.
- The **new row position**, which determines where the new row should be inserted based on the [newRowPosition](#) property.

In the tree grid, the `newRowPosition` property offers the following positions to add the row:

- **Top** : When you set `newRowPosition` to **Top**, the new row will be positioned at the top of the Tree Grid.

- **Bottom** : When you set `newRowPosition` to **Bottom**, the new row will be positioned at the bottom of the Tree Grid.
- **Above** : Setting `newRowPosition` to **Above** allows you to position the new row above a specified target row. This is particularly useful when you want to insert a row above an existing one.
- **Below** : Setting `newRowPosition` to **Below** allows you to position the new row below a specified target row. This is beneficial for inserting a row immediately after another.
- **Child** : Selecting **Child** for `newRowPosition` designates the new row as a child of a specified parent row. This is useful for creating hierarchical structures within the Tree Grid, where the new row becomes a subordinate of the specified parent row.

Here's an example of how to add a new row using the `addRecord` method:

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService, EditService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild, ViewEncapsulation } from '@angular/core';
import {
  EditSettingsModel,
  TreeGridComponent,
} from '@syncfusion/ej2-angular-treegrid';
import { RowDataBoundEventArgs } from '@syncfusion/ej2-grids';
import { sampleData } from '../datasource';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService, EditService],
  standalone: true,
  selector: 'app-container',
  template: `<div style="padding:0px 0px 20px 0px">
    <button ej-button id='above' (click)='addabove()'>Add New
    Row as Above position</button>
    <button ej-button id='below' (click)='addbelow()'>Add New
    Row as Below position</button>
    <button ej-button id='child' (click)='addchild()'>Add New
    Row as Child position</button>
  </div>
  <ej-treegrid #treegrid [dataSource]='data'
  [treeColumnIndex]='1' height='250' childMapping='subtasks'
  [editSettings]='editSettings' >
    <e-columns>
```

```

        <e-column field='taskID' headerText='Task ID'
textAlign='Right' isPrimaryKey="true" width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`,
    })
export class AppComponent implements OnInit {
    public data?: Object[];
    public editSettings?: EditSettingsModel;
    @ViewChild('treegrid')
    public treegrid: TreeGridComponent | undefined;
    ngOnInit(): void {
        this.data = sampleData;
        this.editSettings = {
            allowEditing: true,
            allowAdding: true,
            allowDeleting: true,
        };
    }
    addabove() {
        const newRecord = {
            taskID: this.generatetaskID(),
            taskName: this.generatetaskName(),
            startDate: this.generatestartDate(),
            duration: this.generateduration(),
        };
        (this.treegrid as TreeGridComponent).addRecord(newRecord, 0, 'Above');
    }
    addbelow() {
        const newRecord = {
            taskID: this.generatetaskID(),
            taskName: this.generatetaskName(),
            startDate: this.generatestartDate(),
            duration: this.generateduration(),
        };
        (this.treegrid as TreeGridComponent).addRecord(newRecord, 4, 'Below');
    }
    addchild() {
        const newRecord = {
            taskID: this.generatetaskID(),
            taskName: this.generatetaskName(),
            startDate: this.generatestartDate(),
            duration: this.generateduration(),
        };
        (this.treegrid as TreeGridComponent).clearSelection();
        (this.treegrid as TreeGridComponent).addRecord(newRecord, 8, 'Child');
    }
    // Generate a random taskID
    generatetaskID(): number {
        return Math.floor(1000 + Math.random() * 90000);
    }
    // Generate a random taskName

```

```

generatetaskName(): string {
    const characters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    let result = '';
    for (let i = 0; i < 5; i++) {
        result += characters.charAt(
            Math.floor(Math.random() * characters.length)
        );
    }
    return result;
}
// Generate a random startDate
generatestartDate(): Date {
    return new Date();
}
// Generate a random duration value
generateduration(): number {
    return Math.random() * 100;
}
}

```

MAIN.TS

```

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

```

* If you want to add a new record to the beginning of the data source, you can pass **0** as the second parameter to the `addRecord` method.

* If you do not specify an index, the new row will be added at the end of the tree grid.

Show or hide a row using an external actions

In a Syncfusion tree grid, you can show or hide a particular row based on some external action, such as a checkbox click. This can be useful in scenarios where you want to hide certain rows from the tree grid temporarily, without removing them from the underlying data source. This can be achieved by using the `getRowByIndex` method of the treegrid and `getRowsObject` method of the grid object within the tree grid instance along with the `change` event of the checkbox.

The `getRowsObject` method returns an array of row objects that represents all the rows in the tree grid. You can use this method to iterate through all the rows and access their data and index.

The `getRowByIndex` method returns the HTML element of a row at the specified index. You can use this method to get a specific row and apply changes to it.

In the following example, the `onCheckBoxChange` method is used to check whether the checkbox is checked or not. If it is checked, the method iterates through all the rows in the tree grid using the `getRowsObject` method. For each row, it checks whether the value in the **Duration** column is equal to '0'. If it is, the index of that row is obtained using the `getRowByIndex` method and hidden by setting its display style to "none". The index of the hidden row is also added to an array called `hiddenRows`.

If the checkbox is unchecked, the method iterates through the `hiddenRows` array and shows each row by setting its display style to an empty string. The `hiddenRows` array is also cleared.

APP.COMPONENT.TS

```

{% raw %}
import { Component, OnInit, ViewChild, ViewEncapsulation } from
 '@angular/core';
import {
  EditSettingsModel,
  TreeGridComponent,
} from '@syncfusion/ej2-angular-treegrid';
import { RowDataBoundEventArgs } from '@syncfusion/ej2-grids';
import { sampleData } from '../datasource';
import { ChangeEventArgs } from '@syncfusion/ej2-buttons';
@Component({
  selector: 'app-container',
  encapsulation: ViewEncapsulation.None,
  template: `<div style="padding:2px 0px 0px 0px">
    <ejs-checkbox #checkbox label='Show / Hide Row'
    (change)="onCheckBoxChange($event)"></ejs-checkbox>
  </div>
  <div><p id="message" style="color:red; text-align:center;font-weight:
  bold;">{{ message }}</p> </div>
  <ejs-treegrid #treegrid [dataSource]='data' [treeColumnIndex]='1'
  height='315' childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID' textAlign='Right'
      isPrimaryKey="true" width=90></e-column>
      <e-column field='taskName' headerText='Task Name' textAlign='Left'
      width=180></e-column>
      <e-column field='startDate' headerText='Start Date' textAlign='Right'
      format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration' textAlign='Right'
      width=80></e-column>
    </e-columns>
  </ejs-treegrid>`,
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public rowIndex?: number;
  public hiddenRows: number[] = [];
  @ViewChild('treegrid')
  public treegrid: TreeGridComponent | undefined;
  public message?: string = '';
  ngOnInit(): void {
    this.data = sampleData;
  }
  public onCheckBoxChange(args: ChangeEventArgs) {
    if (args.checked) {
      for (let i = 0; i < (this.treegrid as
      TreeGridComponent).grid.getRowsObject().length;i++) {
        if (((this.treegrid as TreeGridComponent).grid.getRowsObject()[i].data as
        any).duration === 0
        ) {
          // check the row value
          this.rowIndex = (this.treegrid as
          TreeGridComponent).grid.getRowsObject()[i].index;
          //get particular row index

```

```

((this.treegrid as TreeGridComponent).getRowByIndex(this.rowIndex) as
HTMLElement).style.display = 'none'; //hide row
this.hiddenRows.push(this.rowIndex as number); // add row index to
hiddenRows array
}
}
if (this.hiddenRows.length > 0) {
this.message = `Rows with a duration column value of '0' have been hidden`;
}
} else {
// Show hidden rows
this.hiddenRows.forEach((rowIndex: number) => {
((this.treegrid as TreeGridComponent).getRowByIndex(rowIndex) as
HTMLElement).style.display = '';
});
this.hiddenRows = [];
this.message = 'Show all hidden rows';
}
}
}
{% enddraw %}

```

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 get the row data and element

The Tree Grid provides several methods to retrieve row data and elements. This feature is useful when you need to access specific rows, perform custom operations, or manipulate the data displayed in the grid.

1. [getRowByIndex](#): This method returns the HTML element of a row at the specified index. It can be used to retrieve the element of a specific row in the tree grid.

```
`ts
```

```
const rowElement = this.treegrid.getRowByIndex(rowIndex);
```

```
,
```

2. [getRowInfo](#): This method allows you to retrieve row information based on a cell target element.

```
`ts
```

```
const rowInformation = this.treegrid.getRowInfo(targetElement);
```

```
,
```

3. [getRows](#): This method returns an array of all the row elements in the Tree Grid. If you need to retrieve row data and elements, you can combine the `getRows` method with the `getRowInfo` method.

```
`ts
```

```
const rowElements = this.treegrid.getRows();
```

```
,
```

4. [getSelectedRowIndexes](#): This method allows you to retrieve the collection of indexes of the selected rows. However, it does not directly provide the row elements and associated data. To access the row elements and data of the selected rows, you can combine the `getSelectedRowIndexes` method with `getRowByIndex` and `getRowInfo` method.

```
`ts
```

```
const selectedIndexes = this.treegrid.getSelectedRowIndexes();
```

```
,
```

5. [getSelectedRows](#): This method returns an array of HTML elements representing the selected rows in the tree grid. By iterating over this array, you can access each row element and data using the `getRowInfo` method. This way, you can access both the row elements and their associated data for the selected rows.

```
`ts
```

```
const selectedRowElements = this.treegrid.getSelectedRows();
```

```
,
```

See also

- [Render parent rows in collapsed state](#)
- [Retain expanded and collapsed state](#)
- [Persist expanded collapsed state on page refresh using localStorage](#)
- [Programmatically expand and collapse a row](#)
- [Expand and Collapse action events](#)

Cell

Sorting in Angular TreeGrid component

The TreeGrid component provides built-in support for sorting data-bound columns in ascending or descending order. To enable sorting in the tree grid, set the [allowSorting](#) property to **true**.

In the TreeGrid component, sorting follows a specific order to maintain the hierarchical structure of the data. When sorting is performed, the parent records from all rows are sorted initially. Following this, the child records within each parent are sorted internally based on the specified sorting order. This sorting process occurs within the hierarchy order once the parent records are sorted.

This behavior ensures that the hierarchical relationship between parent and child records is preserved, and child records are sorted relative to their respective parent records.

To sort a particular column in the tree grid, click on its column header. Each time you click the header, the order of the column will switch between **Ascending** and **Descending**.

To use the sorting feature, you need to inject the **SortService** in the provider section of **AppModule**.

The following demo illustrates how to enable sorting in the tree grid.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { PageService, SortService,
FilterService, ToolbarService, TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit } from '@angular/core';
import { sortData } from './datasource';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' height='250'
[treeColumnIndex]='1' [allowSorting]='true' childMapping='subtasks' >
    <e-columns>
      <e-column field='Category' headerText='Category'
textAlign='Right' width=140></e-column>
      <e-column field='orderName' headerText='Order Name'
textAlign='Left' width=200></e-column>
      <e-column field='orderDate' headerText='Order Date'
textAlign='Right' format='yMd' width=150></e-column>
      <e-column field='units' headerText='Units'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  ngOnInit(): void {
    this.data = sortData;
  }
}
```

MAIN.TS

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

Initial sorting

By default, the TreeGrid component does not apply any sorting to its records at initial rendering.

However, you can apply initial sorting by setting the [sortSettings.columns](#) property to the desired [field](#) and sort [direction](#). This feature is helpful when you want to display your data in a specific order when the tree grid is first loaded.

The following example demonstrates how to set `sortSettings.columns` for **Category** and **orderName** columns with a specified `direction`.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { PageService, SortService,
FilterService, ToolbarService, TreeGridModule } from '@syncfusion/ej2-angular-
treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit } from '@angular/core';
import { sortData } from './datasource';
@Component({
imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
],
providers: [PageService,
    SortService,
    FilterService,
    ToolbarService],
standalone: true,
selector: 'app-container',
template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='250' [allowSorting]='true' childMapping='subtasks'
[sortSettings]='initialSort'>
    <e-columns>
        <e-column field='Category' headerText='Category'
textAlign='Right' width=140></e-column>
        <e-column field='orderName' headerText='Order Name'
textAlign='Left' width=200></e-column>
        <e-column field='orderDate' headerText='Order Date'
textAlign='Right' format='yMd' width=150></e-column>
        <e-column field='units' headerText='Units'
textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`
})
export class AppComponent implements OnInit {
    public data?: Object[];
```

```

    public initialSort?: Object;
    ngOnInit(): void {
        this.data = sortData;
        this.initialSort = { columns: [{ field: 'Category', direction:
'Ascending' },
                                     { field: 'orderName', direction:
'Ascending' }] }];
    }
}

```

MAIN.TS

```

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

```

The initial sorting defined in `sortSettings.columns` will override any sorting applied through interaction.

Multi-column sorting

The TreeGrid component allows to sort more than one column at a time using multi-column sorting. To enable multi-column sorting in the tree grid, set the `allowSorting` property to **true**, and set the `allowMultiSorting` property to **true** which enable to sort multiple columns by hold the **CTRL** key and click on the column headers. This feature is useful when you want to sort your data based on multiple criteria to analyze it in various ways.

To clear multi-column sorting for a particular column, press the "Shift + mouse left click".

* The `allowSorting` must be true while enabling multi-column sort.

* Set `allowMultiSorting` property as **false** to disable multi-column sorting.

The following demo illustrates how to enable multi-column sorting in the tree grid.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { PageService, SortService,
FilterService, ToolbarService, TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit } from '@angular/core';
import { sortData } from './datasource';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService,

```

```

        ToolbarService],
standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid [dataSource]='data' height='250'
[treeColumnIndex]='1' [allowMultiSorting]='true' [allowSorting]='true'
childMapping='subtasks' >
        <e-columns>
            <e-column field='Category' headerText='Category'
textAlign='Right' width=140></e-column>
            <e-column field='orderName' headerText='Order Name'
textAlign='Left' width=200></e-column>
            <e-column field='orderDate' headerText='Order Date'
textAlign='Right' format='yMd' width=150></e-column>
            <e-column field='units' headerText='Units'
textAlign='Right' width=80></e-column>
        </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
    public data?: Object[];
    ngOnInit(): void {
        this.data = sortData;
    }
}

```

MAIN.TS

```

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

```

Sort multiple columns without holding the CTRL key

You can perform multiple columns sorting without holding the CTRL key in the tree grid. This action can be achieved by enabling the `enableSortMultiTouch` property of the Sort module of the grid object, which can be accessed using the tree grid instance. This `enableSortMultiTouch` property can be enabled within the `created` event of the tree grid.

The following example demonstrates how to enable the `enableSortMultiTouch` property inside the tree grid's `created` event:

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { PageService, SortService,
FilterService, ToolbarService, TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { sortData } from './datasource';
@Component({
imports: [

```

```

        TreeGridModule,
        ButtonModule,
        DropDownListAllModule
    ],
    providers: [PageService,
                SortService,
                FilterService,
                ToolbarService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid #treegrid [dataSource]='data' height='250'
[treeColumnIndex]='1' [allowSorting]='true' childMapping='subtasks'
(created)="created($event)" >
        <e-columns>
            <e-column field='Category' headerText='Category'
textAlign='Right' width=140></e-column>
            <e-column field='orderName' headerText='Order Name'
textAlign='Left' width=200></e-column>
            <e-column field='orderDate' headerText='Order Date'
textAlign='Right' format='yMd' width=150></e-column>
            <e-column field='units' headerText='Units'
textAlign='Right' width=80></e-column>
        </e-columns>
    </ejs-treegrid>`,
    ))
export class AppComponent implements OnInit {
    public data?: Object[];
    @ViewChild('treegrid')
    public treegrid: TreeGridComponent | undefined;
    ngOnInit(): void {
        this.data = sortData;
    }
    created(args: any) {
        (
            (this.treegrid as TreeGridComponent).grid.sortModule as any
        ).enableSortMultiTouch = true;
    }
}

```

MAIN.TS

```

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

```

Prevent sorting for particular column

The TreeGrid component provides the ability to prevent sorting for a particular column. This can be useful when you have certain columns that you do not want to be included in the sorting process.

It can be achieved by setting the [allowSorting](#) property of the particular column to **false**.

The following example demonstrates, how to disable sorting for **orderName** column.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { PageService, SortService,
FilterService, ToolbarService, TreeGridModule } from '@syncfusion/ej2-angular-
treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit } from '@angular/core';
import { sortData } from './datasource';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='250' [allowSorting]='true' childMapping='subtasks' >
    <e-columns>
      <e-column field='Category' headerText='Category'
textAlign='Right' width=140></e-column>
      <e-column field='orderName' headerText='Order Name'
[allowSorting]=false textAlign='Left' width=200></e-column>
      <e-column field='orderDate' headerText='Order Date'
textAlign='Right' format='yMd' width=150></e-column>
      <e-column field='units' headerText='Units'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: object[];
  ngOnInit(): void {
    this.data = sortData;
  }
}

```

MAIN.TS

```

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

```

Sort order

By default, the sorting order will be as **ascending -> descending -> none**.

When you click on a column header for the first time, it sorts the column in ascending order. Clicking the same column header again will sort the column in descending order. A repetitive third click on the same column header will clear the sorting.

Custom sorting

The TreeGrid component provides a way to customize the default sort action for a column by defining the [column.sortComparer](#) property. The sort comparer function works similar to the [Array.sort](#) comparer function, and allows to define custom sorting logic for a specific column.

In the following example, the custom sort comparer function was defined in the **orderName** column.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { PageService, SortService,
FilterService, ToolbarService, TreeGridModule } from '@syncfusion/ej2-angular-
treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit } from '@angular/core';
import { sortData } from './datasource';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='250' [allowSorting]='true' childMapping='subtasks' >
    <e-columns>
      <e-column field='Category' headerText='Category'
textAlign='Right' width=140></e-column>
      <e-column field='orderName'
[sortComparer]='sortComparer' headerText='Order Name' textAlign='Left'
width=200></e-column>
      <e-column field='orderDate' headerText='Order Date'
textAlign='Right' format='yMd' width=150></e-column>
      <e-column field='units' headerText='Units'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data: Object[] = [];
  public sortComparer = (reference: string, comparer: string) => {
    if (reference < comparer) {
      return -1;
    }
    if (reference > comparer) {
```

```

        return 1;
    }
    return 0;
};
ngOnInit(): void {
    this.data = sortData;
}
}

```

MAIN.TS

```



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

```

The customSortComparer function takes two parameters: a and b. The a and b parameters are the values to be compared. The function returns -1, 0, or 1, depending on the comparison result.

Touch interaction

When you tap tree grid header on touch screen devices, then the selected column header is sorted and

display a popup  for multi-column sorting, tap on the popup to sort multiple columns  and then tap the desired tree grid headers.

The [allowMultiSorting](#) and [allowSorting](#) should be **true** then only the popup will be shown.

The following screenshot represents a tree grid touch sorting in the device.

<!-- markdownlint-disable MD033 -->

<!-- markdownlint-enable MD033 -->

Perform sorting based on its culture

Sorting based on culture in the tree grid can be achieved by utilizing the [locale](#) property. By setting the [locale](#) property to the desired culture code, you enable sorting based on that specific culture. This allows you to apply locale-specific sorting rules and ensure accurate ordering for different languages and regions.

In the following example, sorting is performed based on the “ar” locale using the [column.sortComparer](#) property.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { PageService, SortService,
FilterService, ToolbarService, TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild, ViewEncapsulation } from '@angular/core';

```



```

import { DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { SortDirection, SortEventArgs } from '@syncfusion/ej2-grids';
import {
    L10n,
    loadCldr,
    setCulture,
    setCurrencyCode,
} from '@syncfusion/ej2-base';
import { sortData } from './datasource';
import * as cagregorian from './ca-gregorian.json';
import * as currencies from './currencies.json';
import * as numbers from './numbers.json';
import * as timeZoneNames from './timeZoneNames.json';
import * as numberingSystems from './numberingSystems.json';
@Component({
    imports: [

        TreeGridModule,
        ButtonModule,
        DropDownListAllModule
    ],
    providers: [PageService,
        SortService,
        FilterService,
        ToolbarService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid #treegrid [dataSource]='data' height='250'
    [treeColumnIndex]='1' [allowSorting]='true' locale='ar'
    childMapping='subtasks' >
        <e-columns>
            <e-column field='Category' headerText='Category'
    textAlign='Right' width=140></e-column>
            <e-column field='orderName' headerText='Order Name'
    textAlign='Left' width=200></e-column>
            <e-column field='orderDate' headerText='Order Date'
    [sortComparer]="sortComparer" textAlign='Right' format='yMd' width=150></e-
    column>
            <e-column field='price' headerText='Price' format='C2'
    type="number" [sortComparer]="sortComparer" textAlign='Right' width=80></e-
    column>
        </e-columns>
    </ejs-treegrid>`,
})
export class AppComponent implements OnInit {
    public data?: Object[];
    public initialSort?: Object;
    public message?: string;
    @ViewChild('treegrid')
    public treegrid?: TreeGridComponent;
    public formatOptions?: object;
    ngOnInit(): void {
        this.data = sortData;
        setCulture('ar');
        setCurrencyCode('QAR');
    }
}

```

```

    loadCldr(cagregorian, currencies, numbers, timeZoneNames,
    numberingSystems);
    this.formatOptions = { type: 'date', format: 'yyyy/MMM/dd' };
  }
  public sortComparer = (
    reference: number | Date | string,
    comparer: number | Date | string,
    sortOrder: 'Ascending' | 'Descending'
  ) => {
    const referenceDate = new Date(reference);
    const comparerDate = new Date(comparer);
    if (typeof reference === 'number' && typeof comparer === 'number') {
      // Numeric column sorting
      return sortOrder === 'Ascending'
        ? comparer - reference
        : reference - comparer;
    } else if (
      !isNaN(referenceDate.getTime()) &&
      !isNaN(comparerDate.getTime())
    ) {
      // Date column sorting
      return sortOrder === 'Ascending'
        ? comparerDate.getTime() - referenceDate.getTime()
        : referenceDate.getTime() - comparerDate.getTime();
    } else {
      // Default sorting for other types
      const intlCollator = new Intl.Collator(undefined, {
        sensitivity: 'variant',
        usage: 'sort',
      });
      const comparisonResult = intlCollator.compare(
        String(reference),
        String(comparer)
      );
      return sortOrder === 'Ascending' ? -comparisonResult :
      comparisonResult;
    }
  };
}

```

MAIN.TS

```

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

```

Customize sort icon

To customize the sort icon in the tree grid, you can override the default tree grid classes **.e-icon-ascending** and **.e-icon-descending** with custom content using CSS. Simply specify the desired icons or symbols using the **content** property as mentioned below

```
`css
```

```
.e-treegrid .e-icon-ascending::before {
```

```

content: '\e306';
}
.e-treegrid .e-icon-descending::before {
content: '\e304';
}
`

```

In the below sample, tree grid is rendered with a customized sort icon.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { PageService, SortService,
FilterService, ToolbarService, TreeGridModule } from '@syncfusion/ej2-angular-
treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import { sortData } from './datasource';
@Component({
imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
],
providers: [PageService,
            SortService,
            FilterService,
            ToolbarService],
standalone: true,
selector: 'app-container',
encapsulation: ViewEncapsulation.None,
template: `<ejs-treegrid [dataSource]='data' height='250'
[treeColumnIndex]='1' [allowSorting]='true' childMapping='subtasks'
[sortSettings]='initialSort'>
    <e-columns>
        <e-column field='Category' headerText='Category'
textAlign='Right' width=140></e-column>
        <e-column field='orderName' headerText='Order Name'
textAlign='Left' width=200></e-column>
        <e-column field='orderDate' headerText='Order Date'
textAlign='Right' format='yMd' width=150></e-column>
        <e-column field='units' headerText='Units'
textAlign='Right' width=80></e-column>
    </e-columns>
    </ejs-treegrid>`,
styleUrls: ['sorting_style.css']
})
export class AppComponent implements OnInit {
    public data?: Object[];
    public initialSort?: Object;
    ngOnInit(): void {

```

```

    this.data = sortData;
    this.initialSort = {
      columns: [
        { field: 'Category', direction: 'Ascending' },
        { field: 'orderName', direction: 'Ascending' },
      ],
    };
  }
}

```

SORTING STYLE.CSS

```

.e-treegrid .e-icon-ascending::before {
  content: '\e306' !important;
}

.e-treegrid .e-icon-descending::before {
  content: '\e304';
}

```

MAIN.TS

```

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

```

Sort columns externally

The TreeGrid component in Syncfusion's Angular suite allows you to customize the sorting of columns and provides flexibility in sorting based on external interactions. You can sort columns, remove a sort column, and clear sorting using an external button click.

Add sort columns

To sort a column externally, you can utilize the [sortByColumn](#) method with parameters **columnName**, **direction** and **isMultiSort** provided by the TreeGrid component. This method allows you to programmatically sort a specific column based on your requirements.

The following example demonstrates how to add sort columns to a tree grid. It utilizes the **DropDownList** component to select the column and sort direction. When an external button is clicked, the `sortByColumn` method is called with the specified **columnName**, **direction**, and **isMultiSort** parameters.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { PageService, SortService,
FilterService, ToolbarService, TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild, ViewEncapsulation } from '@angular/core';
import { DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns';

```

```

import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { SortDirection } from '@syncfusion/ej2-grids';
import { sortData } from './datasource';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `
    <div style="display: flex">
      <label style="padding: 30px 20px 0 0"> Column name :</label>
      <ejs-dropdownlist #dropdownColumn style="padding: 26px 0 0 0" index="0" width="120" [dataSource]="columns" [fields]="field"></ejs-dropdownlist>
    </div>
    <div style="display: flex">
      <label style="padding: 30px 17px 0 0"> Sorting direction
    :</label>
      <ejs-dropdownlist #dropdownDirection style="padding: 26px 0 0 0" index="0" width="120" [dataSource]="direction"></ejs-dropdownlist>
    </div>
    <button style="margin-top: 10px " ej-button id="button"
    cssClass="e-outline" (click)="addSortColumn()">Add sort column</button>

    <ejs-treegrid #treegrid [dataSource]='data' height='250'
    [treeColumnIndex]='1' [allowSorting]='true' childMapping='subtasks'
    [sortSettings]='initialSort'>
      <e-columns>
        <e-column field='Category' headerText='Category'
        textAlign='Right' width=140></e-column>
        <e-column field='orderName' headerText='Order Name'
        textAlign='Left' width=200></e-column>
        <e-column field='orderDate' headerText='Order Date'
        textAlign='Right' format='yMd' width=150></e-column>
        <e-column field='units' headerText='Units'
        textAlign='Right' width=80></e-column>
      </e-columns>
    </ejs-treegrid>`,
  })
export class AppComponent implements OnInit {
  public data?: Object[];
  public initialSort?: Object;
  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  @ViewChild('dropdownColumn')
  public dropDownColumn?: DropDownListComponent;
  @ViewChild('dropdownDirection')
  public dropDownDirection?: DropDownListComponent;
  public columns: object[] = [

```

```

    { text: 'Category', value: 'Category' },
    { text: 'Order Date', value: 'orderDate' },
    { text: 'Units', value: 'units' },
  ];
  public direction: object[] = [
    { text: 'Ascending', value: 'Ascending' },
    { text: 'Descending', value: 'Descending' },
  ];
  public field: object = { text: 'text', value: 'value' };
  ngOnInit(): void {
    this.data = sortData;
    this.initialSort = {
      columns: [{ field: 'orderName', direction: 'Ascending' }],
    };
  }
  addSortColumn() {
    (this.treegrid as TreeGridComponent).sortByColumn(
      (this.dropDownColumn as DropDownListComponent).value as SortDirection,
      (this.dropDownDirection as DropDownListComponent).value as
SortDirection,
      true
    );
  }
}

```

MAIN.TS

```

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

```

Remove sort columns

To remove a sort column externally, you can use the `removeSortColumn` method provided by the TreeGrid component. This method allows you to remove the sorting applied to a specific column.

The following example demonstrates how to remove sort columns. It utilizes the **DropDownList** component to select the column. When an external button is clicked, the `removeSortColumn` method is called to remove the selected sort column.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { PageService, SortService,
FilterService, ToolbarService, TreeGridModule } from '@syncfusion/ej2-angular-
treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild, ViewEncapsulation } from
'@angular/core';
import { DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { SortDirection } from '@syncfusion/ej2-grids';
import { sortData } from './datasource';

```

```

@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `
    <div style="display: flex">
      <label style="padding: 30px 20px 0 0"> Column name :</label>
      <ejs-dropdownlist #dropdown style="padding: 26px 0 0 0"
index="0" width="120" [dataSource]="columns" [fields]="field"></ejs-
dropdownlist>
    </div>
    <button style="margin-top: 10px " ejs-button id="button"
cssClass="e-outline" (click)="removeSortColumn()" >Remove sort
column</button>

    <ejs-treegrid #treegrid [dataSource]='data' height='250'
[treeColumnIndex]='1' [allowSorting]='true' childMapping='subtasks'
[sortSettings]='initialSort'>
      <e-columns>
        <e-column field='Category' headerText='Category'
textAlign='Right' width=140></e-column>
        <e-column field='orderName' headerText='Order Name'
textAlign='Left' width=200></e-column>
        <e-column field='orderDate' headerText='Order Date'
textAlign='Right' format='yMd' width=150></e-column>
        <e-column field='units' headerText='Units'
textAlign='Right' width=80></e-column>
      </e-columns>
    </ejs-treegrid>`,
  })
export class AppComponent implements OnInit {
  public data?: Object[];
  public initialSort?: Object;
  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  @ViewChild('dropdown')
  public dropDown?: DropDownListComponent;
  public columns: object[] = [
    { text: 'Category', value: 'Category' },
    { text: 'Order Name', value: 'orderName' },
    { text: 'Order Date', value: 'orderDate' },
    { text: 'Units', value: 'units' },
  ];
  public direction: object[] = [
    { text: 'Ascending', value: 'Ascending' },
    { text: 'Descending', value: 'Descending' },
  ];
  public field: object = { text: 'text', value: 'value' };

```

```

ngOnInit(): void {
    this.data = sortData;
    this.initialSort = {
        columns: [
            { field: 'orderName', direction: 'Ascending' },
            { field: 'Category', direction: 'Descending' },
        ],
    };
}
removeSortColumn() {
    (this.treegrid as TreeGridComponent).removeSortColumn(
        (this.dropDown as DropDownListComponent).value as string
    );
}
}

```

MAIN.TS

```

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

```

Clear sorting

To clear the sorting on an external button click, you can use the [clearSorting](#) method provided by the TreeGrid component. This method clears the sorting applied to all columns in the tree grid.

The following example demonstrates how to clear the sorting using `clearSorting` method in the external button click.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { PageService, SortService,
FilterService, ToolbarService, TreeGridModule } from '@syncfusion/ej2-angular-
treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild, ViewEncapsulation } from
'@angular/core';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { sortData } from './datasource';
@Component({
    imports: [
        TreeGridModule,
        ButtonModule,
        DropDownListAllModule
    ],
    providers: [PageService,
        SortService,
        FilterService,
        ToolbarService],
    standalone: true,

```



```

    selector: 'app-container',
    template: `
      <button ej-button id="button" cssClass="e-outline"
      (click)="onExternalSort()"> Clear Sorting </button>

      <ejs-treegrid #treegrid [dataSource]='data' height='250'
      [treeColumnIndex]='1' [allowSorting]='true' childMapping='subtasks'
      [sortSettings]='initialSort'>
        <e-columns>
          <e-column field='Category' headerText='Category'
          textAlign='Right' width=140></e-column>
          <e-column field='orderName' headerText='Order Name'
          textAlign='Left' width=200></e-column>
          <e-column field='orderDate' headerText='Order Date'
          textAlign='Right' format='yMd' width=150></e-column>
          <e-column field='units' headerText='Units'
          textAlign='Right' width=80></e-column>
        </e-columns>
      </ejs-treegrid>`,
  })
  export class AppComponent implements OnInit {
    public data?: Object[];
    public initialSort?: Object;
    @ViewChild('treegrid')
    public treegrid?: TreeGridComponent;
    ngOnInit(): void {
      this.data = sortData;
      this.initialSort = {
        columns: [
          { field: 'orderName', direction: 'Ascending' },
          { field: 'Category', direction: 'Descending' },
        ],
      };
    }
    onExternalSort() {
      (this.treegrid as TreeGridComponent).clearSorting();
    }
  }

```

MAIN.TS

```

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

```

Sorting events

The TreeGrid component provides two events that are triggered during the sorting action such as [actionBegin](#) and [actionComplete](#). These events can be used to perform any custom actions before and after the sorting action is completed.

1. **actionBegin:** `actionBegin` event is triggered before the sorting action begins. It provides a way to perform any necessary operations before the sorting action takes place. This event provides a

parameter that contains the current tree grid state, including the current sorting column, direction, and data.

2. **actionComplete:** `actionComplete` event is triggered after the sorting action is completed. It provides a way to perform any necessary operations after the sorting action has taken place. This event provides a parameter that contains the current tree grid state, including the sorted data and column information.

The following example demonstrates how the `actionBegin` and `actionComplete` events work when sorting is performed. The `actionBegin` event is used to cancel the sorting of the Category column. The `actionComplete` event is used to display a message.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { PageService, SortService,
FilterService, ToolbarService, TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild, ViewEncapsulation } from '@angular/core';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { sortData } from './datasource';
import { SortEventArgs } from '@syncfusion/ej2-grids';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `
    <div style="margin-left:100px;">
      <p style="color:red;" id="message">{{message}}</p>
    </div>

    <ejs-treegrid #treegrid [dataSource]='data' height='250'
  [treeColumnNameIndex]='1' [allowSorting]='true' childMapping='subtasks'
  (actionComplete)='actionComplete($event)'
  (actionBegin)='actionBegin($event)'>
      <e-columns>
        <e-column field='Category' headerText='Category'
  textAlign='Right' width=140></e-column>
        <e-column field='orderName' headerText='Order Name'
  textAlign='Left' width=200></e-column>
        <e-column field='orderDate' headerText='Order Date'
  textAlign='Right' format='yMd' width=150></e-column>
      </e-columns>
    </ejs-treegrid>
  `
})
export class AppContainerComponent implements OnInit {
  constructor() {}
  ngOnInit() {}
}
```

```

        <e-column field='units' headerText='Units'
textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`,
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public initialSort?: Object;
        public message?: string;
        @ViewChild('treegrid')
        public treegrid?: TreeGridComponent;
        ngOnInit(): void {
            this.data = sortData;
        }
        actionBegin(args: any) {
            if (args.requestType === 'sorting' && args.columnName === 'Category') {
                args.cancel = true;
                this.message = args.requestType + ' action cancelled for ' +
args.columnName + ' column';
            }
        }
        actionComplete({ requestType, columnName }: SortEventArgs) {
            this.message = requestType + ' action completed for ' + columnName + '
column';
        }
    }

```

MAIN.TS

```

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

```

[args.requestType](#) refers to the current action being performed. For example in sorting, the [args.requestType](#) value is **sorting**.

Filtering

Filtering in Angular TreeGrid component

Filtering is a powerful feature in the Syncfusion TreeGrid component that enables you to selectively view data based on specific criteria. It allows you to narrow down large datasets and focus on the information you need, thereby enhancing data analysis and decision-making.

To use filter, inject **FilterService** in the provider section of **AppModule**.

To enable filtering in the tree grid, you need to set the [allowFiltering](#) property of the tree grid to true. Once filtering is enabled, you can configure various filtering options through the [filterSettings](#) property of the tree grid. This property allows you to define the behavior and appearance of the filter.

In the TreeGrid component, filtering actions are executed based on the [filterSettings.hierarchyMode](#) to preserve the hierarchical structure of the data.

When filtering is implemented in a tree grid containing hierarchical data, the filter hierarchy mode dictates how the filtering action propagates throughout the hierarchical structure. This ensures that the filtering operation considers the parent-child relationships within the data.

For further information about the filter hierarchy mode, refer to [this sectionLink to the Video](#).

You can check this video to learn about filtering feature in Angular Tree Grid.

Here is an example that demonstrates the default filtering feature of the tree grid:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='275' [allowFiltering]='true' childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' type='date' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  ngOnInit(): void {
    this.data = sampleData;
  }
}
```

DATASOURCE.TS

```
/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date('02/03/2017'),
    endDate: new Date('02/07/2017'),
```

```

        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
                endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
        ]
    },
    {
        taskID: 12,

```

```

    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
      {
        taskID: 13,
        taskName: 'Phase 1',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'High',
        approved: false,
        progress: 50,
        duration: 11,
        subtasks: [{
          taskID: 14,
          taskName: 'Implementation Module 1',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/27/2017'),
          priority: 'Normal',
          duration: 11,
          progress: 10,
          approved: false,
          subtasks: [
            { taskID: 15, taskName: 'Development Task 1',
              startDate: new Date('02/17/2017'),
              endDate: new Date('02/19/2017'), duration: 3,
              progress: '50', priority: 'High', approved: false },
            { taskID: 16, taskName: 'Development Task 2',
              startDate: new Date('02/17/2017'),
              endDate: new Date('02/19/2017'), duration: 3,
              progress: '50', priority: 'Low', approved: true },
            { taskID: 17, taskName: 'Testing', startDate: new
              Date('02/20/2017'),
              endDate: new Date('02/21/2017'), duration: 2,
              progress: '0', priority: 'Normal', approved: true },
            { taskID: 18, taskName: 'Bug fix', startDate: new
              Date('02/24/2017'),
              endDate: new Date('02/25/2017'), duration: 2,
              progress: '0', priority: 'Critical', approved: false },
            { taskID: 19, taskName: 'Customer review meeting',
              startDate: new Date('02/26/2017'),
              endDate: new Date('02/27/2017'), duration: 2,
              progress: '0', priority: 'High', approved: false },
            { taskID: 20, taskName: 'Phase 1 complete',
              startDate: new Date('02/27/2017'),
              endDate: new Date('02/27/2017'), duration: 0,
              progress: '50', priority: 'Low', approved: true }
          ]
        }
      ]
    },
    {
      taskID: 21,
      taskName: 'Phase 2',

```

```

        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
                { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
                endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
                endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
                { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
                { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
                endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
            ]
        }
    ],
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),

```

```

        priority: 'High',
        approved: false,
        duration: 11,
        progress: 60,
        subtasks: [
            { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
            { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
        ]
    }
}
];

```

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 apply and clear filtering, by using [filterByColumn](#) and [clearFiltering](#) methods.

* To disable Filtering for a particular column, by specifying [columns.allowFiltering](#) to false.

Filter hierarchy modes

The TreeGrid component offers support for a variety of filtering modes through the [filterSettings.hierarchyMode](#) property. By utilizing these filter hierarchy modes, the tree grid offers flexibility in displaying filtered records based on their parent-child relationships, allowing customization of the filtering behavior according to specific requirements.

The following are the types of filter modes available in the Tree Grid:

- **Parent :**
- This is the **default** filter hierarchy mode in tree grid.
- Filtered records are displayed along with their parent records.
- If a filtered record has no parent record, only the filtered record is displayed.
- **Child :**
- Filtered records are displayed along with their child records.
- If a filtered record has no child record, only the filtered record is displayed.
- **Both :**
- Filtered records are displayed along with both their parent and child records.
- If a filtered record has neither parent nor child records, only the filtered record is displayed.
- **None :**
- Only the filtered records are displayed.

The following demo illustrates filtering records with different hierarchy modes.

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ChangeEventArgs } from '@syncfusion/ej2-angular-dropdowns';
import { TreeGridAllModule, TreeGridComponent, FilterService } from
 '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridAllModule, DropDownListAllModule
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<div style="padding-top: 7px; display: inline-
block">Hierarchy Mode
    <ejs-dropdownlist (change)='onChange($event)'
[dataSource]='dropData' value='Parent' [fields]='fields'></ejs-dropdownlist>
    </div>

    <ejs-treegrid #treegrid [dataSource]='data' height='210'
[treeColumnIndex]='1' [allowFiltering]='true' childMapping='subtasks' >
    <e-columns>
      <e-column field='taskId' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' type='date' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public dropData?: Object[];
```

```

public fields?: Object;
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
    this.dropData = [
      { id: 'Parent', mode: 'Parent' },
      { id: 'Child', mode: 'Child' },
      { id: 'Both', mode: 'Both' },
      { id: 'None', mode: 'None' },
    ];
    this.fields = { text: 'mode', value: 'id' };
  }

  onChange(e: ChangeEventArgs): any {
    let mode: any = <string>e.value;
    (this.treeGridObj as TreeGridComponent).filterSettings.hierarchyMode
= mode;
    this.treeGridObj?.clearFiltering();
  }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date('02/03/2017'),
    endDate: new Date('02/07/2017'),
    progress: 100,
    duration: 5,
    priority: 'Normal',
    approved: false,
    subtasks: [
      { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
      { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
      { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
      { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ]
  },
  {

```

```

        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
                priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
                priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
                priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
                priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
                endDate: new Date('02/14/2017'), duration: 0, progress: 0,
                priority: 'Normal', approved: true }
        ],
        {
            taskID: 12,
            taskName: 'Implementation Phase',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            approved: false,
            duration: 11,
            progress: 66,
            subtasks: [
                {
                    taskID: 13,
                    taskName: 'Phase 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'High',
                    approved: false,
                    progress: 50,
                    duration: 11,
                    subtasks: [{
                        taskID: 14,
                        taskName: 'Implementation Module 1',
                        startDate: new Date('02/17/2017'),
                        endDate: new Date('02/27/2017'),
                        priority: 'Normal',
                        duration: 11,

```

```

        progress: 10,
        approved: false,
        subtasks: [
            { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
            { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
            { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
        ]
    }
},
{
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
        taskID: 22,
        taskName: 'Implementation Module 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'Critical',
        approved: false,
        duration: 12,
        progress: 90,
        subtasks: [
            { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
            { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },

```

```

        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  },
  {
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
      taskID: 30,
      taskName: 'Implementation Module 3',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'High',
      approved: false,
      duration: 11,
      progress: 60,
      subtasks: [
        { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
        { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),

```

```

                                endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                                { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
                                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
                                ]
                                }]
                                }
                                ]
                                }
];

```

MAIN.TS

```

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

```

Initial filter

To apply an initial filter, you need to specify the filter criteria using the [predicate](#) object in [filterSettings.columns](#). The `predicate` object represents the filtering condition and contains properties such as field, operator, and value.

Here is an example of how to configure the initial filter using the `predicate` object:

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' height='275'
[treeColumnIndex]='1' [allowFiltering]='true'
[filterSettings]="filterSettings" childMapping='subtasks' >
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>

```

```

        </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public filterSettings?: Object;
        ngOnInit(): void {
            this.data = sampleData;
            this.filterSettings={
                columns: [{ field: 'taskName', matchCase: false, operator:
                'startswith', predicate: 'and', value: 'plan' },
                    { field: 'duration', matchCase: false, operator: 'equal',
                predicate: 'and', value: 5 }]
            };
        }
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
            Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
            priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
            Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
            priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
            Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
            priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
            Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
            priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
    }
]

```

```

        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
              endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
              endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
              endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
              endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
              endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
        ]
    },
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,
                    taskName: 'Implementation Module 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'Normal',
                    duration: 11,
                    progress: 10,
                    approved: false,
                    subtasks: [
                        { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),

```



```

        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
        { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
        { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
      taskID: 22,
      taskName: 'Implementation Module 2',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/28/2017'),
      priority: 'Critical',
      approved: false,
      duration: 12,
      progress: 90,
      subtasks: [
        { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
        { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
          endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },

```

```

        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  }
},
{
  taskID: 29,
  taskName: 'Phase 3',
  startDate: new Date('02/17/2017'),
  endDate: new Date('02/27/2017'),
  priority: 'Normal',
  approved: false,
  duration: 11,
  progress: 30,
  subtasks: [{
    taskID: 30,
    taskName: 'Implementation Module 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'High',
    approved: false,
    duration: 11,
    progress: 60,
    subtasks: [
      { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
      { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
      { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
      { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
      { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
      { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),

```

```

                                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
                                ]
                                }]
                                }
                                ]
                                }
                                ]
];

```

MAIN.TS

```

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

```

Initial filter with multiple values for same column

In the TreeGrid component, you can establish an initial filter containing multiple values for a particular column, which helps you to preset filter conditions for a specific column using multiple values. This functionality allows you to display a filtered records in the tree grid right after the tree grid is initially loaded.

To apply the filter with multiple values for same column at initial rendering, set the filter [predicate](#) object in [filterSettings.columns](#).

The following example demonstrates, how to perform an initial filter with multiple values for same **taskName** column using [filterSettings.columns](#) and [predicate](#).

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit ,ViewChild} from '@angular/core';
import { sampleData } from './datasource';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  standalone: true,
  providers: [FilterService],
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' height='275'
[treeColumnIndex]='1' [allowFiltering]='true'
[filterSettings]="filterSettings" childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  `
})

```

```

        </e-columns>
    </ejs-treegrid>`,
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public filterSettings?: Object;
        ngOnInit(): void {
            this.data = sampleData;
            this.filterSettings = {
                type: 'Excel',
                columns: [
                    {
                        field: 'taskName',
                        matchCase: false,
                        operator: 'startswith',
                        predicate: 'or',
                        value: 'plan',
                    },
                    {
                        field: 'taskName',
                        matchCase: false,
                        operator: 'startswith',
                        predicate: 'or',
                        value: 'design',
                    },
                ],
            };
        }
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),

```

```

        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ]
},
{
    taskID: 6,
    taskName: 'Design',
    startDate: new Date('02/10/2017'),
    endDate: new Date('02/14/2017'),
    duration: 3,
    progress: 86,
    priority: 'High',
    approved: false,
    subtasks: [
        { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
        { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,

```

```

        progress: 50,
        duration: 11,
        subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
                { taskID: 15, taskName: 'Development Task 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
                progress: '50', priority: 'High', approved: false },
                { taskID: 16, taskName: 'Development Task 2',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
                progress: '50', priority: 'Low', approved: true },
                { taskID: 17, taskName: 'Testing', startDate: new
                Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
                progress: '0', priority: 'Normal', approved: true },
                { taskID: 18, taskName: 'Bug fix', startDate: new
                Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
                progress: '0', priority: 'Critical', approved: false },
                { taskID: 19, taskName: 'Customer review meeting',
                startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
                progress: '0', priority: 'High', approved: false },
                { taskID: 20, taskName: 'Phase 1 complete',
                startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
                progress: '50', priority: 'Low', approved: true }
            ]
        }
    ],
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,

```

```

        subtasks: [
            { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
            { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
            { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
            endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
            endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
            { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
            endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
        ]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
                { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),

```

```

        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
  }
}
]
];

```

MAIN.TS

```

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

```

Initial filter with multiple values for different columns

By applying an initial filter with multiple values for different columns in the Tree Grid, you have the flexibility to set predefined filter settings for each column. This results in a filtered records of the tree grid right after the tree grid is initially loaded.

To apply the filter with multiple values for different column at initial rendering, set the filter [predicate](#) object in [filterSettings.columns](#).

The following example demonstrates how to perform an initial filter with multiple values for different **Task ID** and **Task Name** columns using `filterSettings.columns` and `predicate`.

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  standalone: true,
  providers: [FilterService],
  selector: 'app-container',

```



```

    template: `<ejs-treegrid [dataSource]='data' height='275'
[treeColumnIndex]='1' [allowFiltering]='true'
[filterSettings]="filterSettings" childMapping='subtasks' >
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`,
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public filterSettings?: Object;
        ngOnInit(): void {
            this.data = sampleData;
            this.filterSettings = {
                type: 'Excel',
                columns: [
                    {
                        field: 'taskName',
                        matchCase: false,
                        operator: 'startswith',
                        predicate: 'or',
                        value: 'plan',
                    },
                    {
                        field: 'taskName',
                        matchCase: false,
                        operator: 'startswith',
                        predicate: 'or',
                        value: 'design',
                    },
                    {
                        field: 'taskID',
                        matchCase: false,
                        operator: 'lessThan',
                        predicate: 'and',
                        value: 5,
                    },
                    {
                        field: 'taskID',
                        matchCase: false,
                        operator: 'equal',
                        predicate: 'or',
                        value: 2,
                    },
                ],
            },
        ];
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        ]
    }
]

```

```

        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
          endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
      ]
    },
    {
      taskID: 12,
      taskName: 'Implementation Phase',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'Normal',
      approved: false,
      duration: 11,
      progress: 66,
      subtasks: [
        {
          taskID: 13,
          taskName: 'Phase 1',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/27/2017'),
          priority: 'High',
          approved: false,
          progress: 50,
          duration: 11,
          subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
              { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
              { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
              { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
              { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
              { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),

```

```

        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
    }
},
{
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
        taskID: 22,
        taskName: 'Implementation Module 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'Critical',
        approved: false,
        duration: 12,
        progress: 90,
        subtasks: [
            { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
            { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
            { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
                endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
                endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
            { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
                endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
        ]
    }
    ]
},
{
    taskID: 29,

```

```

        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
                progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
                progress: '50', priority: 'Normal', approved: false },
                { taskID: 33, taskName: 'Testing', startDate: new
                Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
                progress: '0', priority: 'Critical', approved: true },
                { taskID: 34, taskName: 'Bug fix', startDate: new
                Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
                progress: '0', priority: 'High', approved: false },
                { taskID: 35, taskName: 'Customer review meeting',
                startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
                progress: '0', priority: 'Normal', approved: true },
                { taskID: 36, taskName: 'Phase 3 complete',
                startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
                progress: '50', priority: 'Critical', approved: false },
            ]
        }
    ]
}
];

```

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

The Syncfusion TreeGrid component provides various filter operators that can be used to define filter conditions for columns. The filter operator for a column can be defined using the [operator](#) property in the [filterSettings.columns](#) object.

The available operators and its supported data types are,

Operator | Description | Supported Types

a*b | Everything that starts with "a" and ends with "b".

a* | Everything that starts with "a".

*b | Everything that ends with "b".

a | Everything that has an "a" in it.

ab* | Everything that has an "a" in it, followed by anything, followed by a "b", followed by anything.

LIKE filtering

The **LIKE** filter can process single search patterns using the "%" symbol, retrieving values matching the specified patterns. The following Tree Grid features support LIKE filtering on string-type columns:

- Filter Menu
- Filter Bar with the [filterSettings.showFilterBarOperator](#) property enabled on the Tree Grid [filterSettings](#).
- Custom Filter of Excel filter type.

For example:

Operator | Description

%ab% | Returns all the value that are contains "ab" character.

ab% | Returns all the value that are ends with "ab" character.

%ab | Returns all the value that are starts with "ab" character.

By default, the Tree Grid uses different filter operators for different column types. The default filter operator for string type columns is **startsWith**, for numerical type columns is **equal**, and for boolean type columns is also **equal**.

Diacritics filter

The diacritics filter feature in the tree grid is useful when working with text data that includes accented characters (diacritic characters). By default, the tree grid ignores these characters during filtering. However, if you need to consider diacritic characters in your filtering process, you can enable this feature by setting the [filterSettings.ignoreAccent](#) property to true using the [filterSettings](#).

Consider the following sample where the `ignoreAccent` property is set to true in order to include diacritic characters in the filtering process:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit ,ViewChild} from '@angular/core';
```

```

import { diacritics } from './datasource';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='0'
height='275' [allowFiltering]='true'
[filterSettings]='filterSettings' childMapping='Children' >
    <e-columns>
      <e-column field='EmpID' headerText='Employee ID'
textAlign='Right' width=90></e-column>
      <e-column field='Name' headerText='Name'
textAlign='Left' width=180></e-column>
      <e-column field='DOB' headerText='DOB'
textAlign='Right' type='date' format='yMd' width=90></e-column>
      <e-column field='Country' textAlign='Right'
width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public filterSettings?: Object;
  ngOnInit(): void {
    this.data = diacritics;
    this.filterSettings = { ignoreAccent: true };
  }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let diacritics: Object[] = [{
  'Name': 'Aeróbics',
  'Designation': 'Chief Executive Officer',
  'EmployeeID': '1',
  'EmpID': 'EMP001',
  'Address': '507 - 20th Ave. E.Apt. 2A, Seattle',
  'Contact': '(206) 555-9857',
  'Country': 'USA',
  'DOB': new Date('2/15/1963'),

  'Children': [{
    'Name': 'Aerografía en Agua',
    'Designation': 'Vice President',
    'EmployeeID': '2',
    'EmpID': 'EMP004',
    'Address': '722 Moss Bay Blvd., Kirkland',
    'Country': 'USA',
    'Contact': '(206) 555-3412',
    'DOB': new Date('5/20/1971'),
  ]
}

```

```

    'Children': [{
      'Name': 'AerografÃa',
      'Designation': 'Marketing Executive',
      'EmployeeID': '3',
      'EmpID': 'EMP035',
      'Address': '4110 Old Redmond Rd., Redmond',
      'Country': 'USA',
      'Contact': '(206) 555-8122',
      'DOB': new Date('3/19/1966'),
      'Children': [
        {
          'Name': 'Aerodelaje',
          'Designation': 'Sales Representative',
          'EmployeeID': '4',
          'EmpID': 'EMP045',
          'Address': '14 Garrett Hill, London',
          'Country': 'UK',
          'Contact': '(71) 555-4848',
          'DOB': new Date('9/20/1980')
        },
        {
          'Name': 'Ãguilas',
          'Designation': 'Sales Representative',
          'EmployeeID': '5',
          'EmpID': 'EMP091',
          'Address': '4726 - 11th Ave. N.E., Seattle',
          'Country': 'USA',
          'Contact': '(206) 555-1189',
          'DOB': new Date('10/19/1989')
        },
        {
          'Name': 'Ãlbumes de Delta',
          'Designation': 'Sales Representative',
          'EmployeeID': '6',
          'EmpID': 'EMP110',
          'Address': 'Coventry House Miner Rd., London',
          'Country': 'UK',
          'Contact': '(71) 555-3636',
          'DOB': new Date('11/02/1987')
        },
        {
          'Name': 'Ãlbumes de MÃsica',
          'Designation': 'Sales Coordinator',
          'EmployeeID': '7',
          'EmpID': 'EMP131',
          'Address': 'Edgeham Hollow Winchester Way, London',
          'Country': 'UK',
          'Contact': '(71) 555-3636',
          'DOB': new Date('11/06/1990')
        }
      ],
    },
    {
      'Name': 'Alusivos',
    }
  ],

```



```

        'Designation': 'Sales Executive',
        'EmployeeID': '8',
        'EmpID': 'EMP039',
        'Address': '7 Houndstooth Rd., London',
        'Country': 'UK',
        'Contact': '(71) 555-3690',
        'DOB': new Date('02/02/1980'),
        'Children': [
            {
                'Name': 'Aerografía',
                'Designation': 'Sales Representative',
                'EmployeeID': '9',
                'EmpID': 'EMP213',
                'Address': '4726 - 11th Ave. N.E., California',
                'Country': 'USA',
                'Contact': '(206) 555-1989',
                'DOB': new Date('01/21/1986')
            },
            {
                'Name': 'Análisis de Escritura a Mano',
                'Designation': 'Sales Coordinator',
                'EmployeeID': '10',
                'EmpID': 'EMP201',
                'Address': 'Coventry House Miner Rd., London',
                'Country': 'UK',
                'Contact': '(71) 555-2222',
                'DOB': new Date('12/01/1990')
            },
            {
                'Name': 'Aerodelaje',
                'Designation': 'Sales Representative',
                'EmployeeID': '11',
                'EmpID': 'EMP197',
                'Address': '200 Lincoln Ave, Salinas, CA 93901',
                'Country': 'USA',
                'Contact': '(831) 758-7408',
                'DOB': new Date('03/23/1987')
            },
            {
                'Name': 'Álbumes de Delta',
                'Designation': 'Sales Representative',
                'EmployeeID': '12',
                'EmpID': 'EMP167',
                'Address': '200 Lincoln Ave, Salinas, CA 93901',
                'Country': 'USA',
                'Contact': '(831) 758-7368',
                'DOB': new Date('08/09/1989')
            }
        ]
    }
}
}];

```

MAIN.TS

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

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

Filtering with case sensitivity

The Tree Grid provides the flexibility to enable or disable case sensitivity during filtering. This feature is useful when you want to control whether filtering operations should consider the case of characters. It can be achieved by using the [enableCaseSensitivity](#) property within the [filterSettings](#) of the grid object through the tree grid instance.

Below is an example code demonstrating how to enable or disable case sensitivity while filtering:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { SwitchModule } from '@syncfusion/ej2-angular-buttons';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridAllModule, SwitchModule
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<div>
    <label for="unchecked"> Enable Case Sensitivity </label>
    <ejs-switch id="unchecked"
(change)="onToggleCaseSensitive($event)"></ejs-switch>
  </div>

    <ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' height='275' [allowFiltering]='true'
[filterSettings]='filtersettings' childMapping='subtasks' >
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' type='date' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
      </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  @ViewChild('treegrid')
  public treegrid: TreeGridComponent | undefined;
  public data?: Object[];
  public filtersettings?: Object;
  public isCaseSensitive: boolean = false;
```

```

ngOnInit(): void {
    this.data = sampleData;
    this.filterSettings={enableCaseSensitivity: this.isCaseSensitive};
}
onToggleCaseSensitive(args:any): void {
    (this.treegrid?.grid.filterSettings as
any).enableCaseSensitivity=args.checked;
}
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [

```

```

        { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
          endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
        { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
          endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
          endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
      ]
    },
    {
      taskID: 12,
      taskName: 'Implementation Phase',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'Normal',
      approved: false,
      duration: 11,
      progress: 66,
      subtasks: [
        {
          taskID: 13,
          taskName: 'Phase 1',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/27/2017'),
          priority: 'High',
          approved: false,
          progress: 50,
          duration: 11,
          subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
              { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
              { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),

```

```

        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
        { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
                { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
                endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
                endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
            ]
        }
    ]
    }
    ]
}

```

```

        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
          endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
          endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
      ]
    },
    {
      taskID: 29,
      taskName: 'Phase 3',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'Normal',
      approved: false,
      duration: 11,
      progress: 30,
      subtasks: [{
        taskID: 30,
        taskName: 'Implementation Module 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'High',
        approved: false,
        duration: 11,
        progress: 60,
        subtasks: [
          { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
          { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
          { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
          { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
          { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
          { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: 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 different filter dialog for a column

The Tree Grid offers the flexibility to customize filtering behavior for different columns by enabling various types of filters such as **Menu**, **Excel**. This feature allows you to tailor the filtering experience to suit the specific needs of each column in your tree grid. For example, you might prefer a menu-based filter for a task name column, an Excel-like filter for a Start date column.

It can be achieved by adjusting the [column.filter.type](#) property based on your requirements.

Here's an example where the menu filter is enabled by default for all columns, but you can dynamically modify the filter types through a dropdown:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { Column, FilterSettingsModel, FilterType, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { DropDownListComponent, ChangeEventArgs } from '@syncfusion/ej2-angular-dropdowns';

@Component({
  imports: [
    TreeGridAllModule, DropDownListAllModule
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<div id="content" class="container">
    <div class="input-container">
      <label for="fields" class="label">Select Column</label>
      <ejs-dropdownlist #field id="fields"
[dataSource]="fieldData" (change)="onFieldChange($event)"
placeholder="Eg: OrderID"></ejs-dropdownlist>
    </div>
    <div class="input-container">
      <label for="types" class="label">Select Filter
Type</label>
      <ejs-dropdownlist #type id="types"
[dataSource]="typeData" (change)="onTypeChange($event)"
placeholder="Eg: Excel" [enabled]="false"></ejs-
dropdownlist>
```

```

        </div>
    </div>

    <ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' height='275' [allowFiltering]='true'
(dataBound)="dataBound()" [filterSettings]='filterSettings'
childMapping='subtasks' >
        <e-columns>
            <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
            <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
            <e-column field='startDate' headerText='Start Date'
textAlign='Right' type='date' format='yMd' width=90></e-column>
            <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
        </e-columns>
    </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        @ViewChild('treegrid')
        public treegrid:TreeGridComponent | undefined;

        @ViewChild('type') public typeDropdown?: DropDownListComponent;
        public data?: Object[];
        public filterSettings?: FilterSettingsModel = { type: 'Menu' };
        public columnFilterSettings?: FilterSettingsModel;
        public fieldData: string[] | undefined;
        public typeData: string[] = [];
        public column_inx: any;
        ngOnInit(): void {
            this.data = sampleData;
        }
        dataBound() {
            this.fieldData = (this.treegrid as
TreeGridComponent).getColumnFieldNames();
        }
        onFieldChange(args: ChangeEventArgs): void {
            (this.typeDropdown as DropDownListComponent).enabled = true;
            this.typeData = ['Menu', 'Excel'];
            this.column_inx = (this.treegrid as
TreeGridComponent).getColumnIndexByField(args.value as string);
        }
        onTypeChange(args: ChangeEventArgs): void {
            this.columnFilterSettings = { type: args.value as FilterType};
            ((this.treegrid as TreeGridComponent).columns[this.column_inx] as
Column).filter=this.columnFilterSettings;
            (this.treegrid as TreeGridComponent).refreshColumns();
        }
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */

```



```

export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date('02/03/2017'),
    endDate: new Date('02/07/2017'),
    progress: 100,
    duration: 5,
    priority: 'Normal',
    approved: false,
    subtasks: [
      { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
      { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
      { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
      { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 6,
    taskName: 'Design',
    startDate: new Date('02/10/2017'),
    endDate: new Date('02/14/2017'),
    duration: 3,
    progress: 86,
    priority: 'High',
    approved: false,
    subtasks: [
      { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
      { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
      { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
      { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
      { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),

```

```

        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
    ],
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,
                    taskName: 'Implementation Module 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'Normal',
                    duration: 11,
                    progress: 10,
                    approved: false,
                    subtasks: [
                        { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
                        { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                        { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
                        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
                            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
                    ]
                }
            ]
        }
    ]
}

```

```

    ]
    }],
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/20/2017'), duration: 4,
                    progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/20/2017'), duration: 4,
                    progress: '50', priority: 'Critical', approved: true },
                { taskID: 25, taskName: 'Testing', startDate: new
                    Date('02/21/2017'),
                    endDate: new Date('02/24/2017'), duration: 2,
                    progress: '0', priority: 'High', approved: false },
                { taskID: 26, taskName: 'Bug fix', startDate: new
                    Date('02/25/2017'),
                    endDate: new Date('02/26/2017'), duration: 2,
                    progress: '0', priority: 'Low', approved: false },
                { taskID: 27, taskName: 'Customer review meeting',
                    startDate: new Date('02/27/2017'),
                    endDate: new Date('02/28/2017'), duration: 2,
                    progress: '0', priority: 'Critical', approved: true },
                { taskID: 28, taskName: 'Phase 2 complete',
                    startDate: new Date('02/28/2017'),
                    endDate: new Date('02/28/2017'), duration: 0,
                    progress: '50', priority: 'Normal', approved: false }
            ]
        }
    ]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
    }
]

```

```

        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'Normal', approved: false },
                { taskID: 33, taskName: 'Testing', startDate: new
                    Date('02/20/2017'),
                    endDate: new Date('02/21/2017'), duration: 2,
                    progress: '0', priority: 'Critical', approved: true },
                { taskID: 34, taskName: 'Bug fix', startDate: new
                    Date('02/24/2017'),
                    endDate: new Date('02/25/2017'), duration: 2,
                    progress: '0', priority: 'High', approved: false },
                { taskID: 35, taskName: 'Customer review meeting',
                    startDate: new Date('02/26/2017'),
                    endDate: new Date('02/27/2017'), duration: 2,
                    progress: '0', priority: 'Normal', approved: true },
                { taskID: 36, taskName: 'Phase 3 complete',
                    startDate: new Date('02/27/2017'),
                    endDate: new Date('02/27/2017'), duration: 0,
                    progress: '50', priority: 'Critical', approved: false },
            ]
        }]
    }
]
};

```

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 view the GitHub sample demonstrating different filter dialogs in a TreeGrid column.](#)

Filter grid programmatically with single and multiple values using method

Programmatic filtering in the Tree Grid with single and multiple values allows you to apply filters to specific columns in the tree grid without relying on interactions through the interface.

This can be achieved by utilizing the [filterByColumn](#) method of the tree grid.

The following example demonstrates, how to programmatically filter the tree grid using single and multiple values for the **taskID** and **taskName** columns. This is accomplished by calling the **filterByColumn** method within an external button click function.

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { FilterSettingsModel, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';

@Component({
  imports: [
    TreeGridAllModule, ButtonModule
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: ` <button ejs-button cssClass="e-outline"
(click)="onSingleValueFilter()">Filter with single value</button>
      <button ejs-button cssClass="e-outline" style="margin-left:5px" (click)="onMultipleValueFilter()">Filter with multiple
values</button>

      <ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' height='275' [allowFiltering]='true'
[filterSettings]='filterSettings' childMapping='subtasks' >
        <e-columns>
          <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
          <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
          <e-column field='startDate' headerText='Start Date'
textAlign='Right' type='date' format='yMd' width=90></e-column>
          <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
        </e-columns>
      </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  @ViewChild('treegrid')
  public treegrid:TreeGridComponent | undefined;
  public data?: Object[];
  public filterSettings?: FilterSettingsModel = { type: 'Excel' };

  ngOnInit(): void {
    this.data = sampleData;
  }

  onSingleValueFilter() {
    (this.treegrid as TreeGridComponent).clearFiltering();
    // filter TaskID column with single value
  }
}
```

```

        (this.treegrid as TreeGridComponent).filterByColumn('taskID', 'equal',
7);
    }
    onMultipleValueFilter() {
        (this.treegrid as TreeGridComponent).clearFiltering();
        // filter TaskName column with multiple values
        (this.treegrid as TreeGridComponent).filterByColumn('taskName',
'equal', [
            'Planning Complete',
            'Testing',
            'Design complete',
        ]);
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
    }
]

```

```

        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
              endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
              endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
              endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
              endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
              endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
        ]
    },
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,
                    taskName: 'Implementation Module 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'Normal',
                    duration: 11,
                    progress: 10,
                    approved: false,
                    subtasks: [
                        { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),

```

```

        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
        { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
        { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
      taskID: 22,
      taskName: 'Implementation Module 2',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/28/2017'),
      priority: 'Critical',
      approved: false,
      duration: 12,
      progress: 90,
      subtasks: [
        { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
        { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
          endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },

```



```

        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  }
},
{
  taskID: 29,
  taskName: 'Phase 3',
  startDate: new Date('02/17/2017'),
  endDate: new Date('02/27/2017'),
  priority: 'Normal',
  approved: false,
  duration: 11,
  progress: 30,
  subtasks: [{
    taskID: 30,
    taskName: 'Implementation Module 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'High',
    approved: false,
    duration: 11,
    progress: 60,
    subtasks: [
      { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
      { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
      { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
      { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
      { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
      { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),

```

```

                                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
                                ]
                                }]
                                }
                                ]
                                }
                                ]
];

```

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 get filtered records

Retrieving filtered records in the tree grid is essential when you want to work with data that matches the currently applied filters. You can achieve this using available properties in the TreeGrid component.

1.Using the Filtered result property

The `filteredResult` property of filter module serves to obtain an array of records that correspond to the currently applied filters on the tree grid.

This property retrieves an array of records that match the currently applied filters on the tree grid by leveraging the filter module.

Below is an example demonstrating how to retrieve filtering data in a tree grid using the `filteredResult` property:

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { FilterSettingsModel, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridAllModule, ButtonModule
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: ` <div class="control-section">
    <div *ngIf="showWarning">
      <ejs-message id="msg_warning" content="No Records"
cssClass="e-content-center"
      severity="Warning"></ejs-message>
    </div>
  `
})

```

```

        <button ejs-button cssClass="e-success"
(click)="click()">Get Filtered Data</button>
        <button ejs-button cssClass="e-danger"
(click)="clear()">Clear</button>
        <ejs-treegrid #treegrid id='treegrid' [dataSource]='data'
[treeColumnIndex]='1' height='275' [allowFiltering]='true'
childMapping='subtasks' >
            <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
                <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
                <e-column field='startDate' headerText='Start Date'
textAlign='Right' type='date' format='yMd' width=90></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
            </e-columns>
        </ejs-treegrid>

        <div *ngIf="showRecords" class="e-content">
            <h3>Filtered Records</h3>
            <ejs-treegrid #filtergrid [dataSource]="filteredData"
childMapping='subtasks' allowPaging="true" [height]="200">
                <e-columns>
                    <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
                    <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
                    <e-column field='startDate' headerText='Start Date'
textAlign='Right' type='date' format='yMd' width=90></e-column>
                    <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
                </e-columns>
            </ejs-treegrid>
        </div>
    </div>`
    })
    export class AppComponent implements OnInit {
        @ViewChild('treegrid')
        public treegrid:TreeGridComponent | undefined;
        public filteredData?: Object;
        public data?: Object[];
        showRecords?: boolean;
        showWarning?: boolean;
        ngOnInit(): void {
            this.data = sampleData;
        }
        click(): void {
            this.filteredData = (this.treegrid as any).filterModule.filteredResult;
            if (this.filteredData) {
                this.showRecords = true;
            } else {
                this.showRecords = false;
            }
            this.showWarning = !this.showRecords;
        }
        clear(): void {

```

```

    (this.treegrid as TreeGridComponent).clearFiltering();
    this.showRecords = false;
    this.showWarning = false;
  }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date('02/03/2017'),
    endDate: new Date('02/07/2017'),
    progress: 100,
    duration: 5,
    priority: 'Normal',
    approved: false,
    subtasks: [
      { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
      { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
      { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
      { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ],
  },
  {
    taskID: 6,
    taskName: 'Design',
    startDate: new Date('02/10/2017'),
    endDate: new Date('02/14/2017'),
    duration: 3,
    progress: 86,
    priority: 'High',
    approved: false,
    subtasks: [
      { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
      { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),

```

```

        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
        priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
        priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            progress: 50,
            duration: 11,
            subtasks: [{
                taskID: 14,
                taskName: 'Implementation Module 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'Normal',
                duration: 11,
                progress: 10,
                approved: false,
                subtasks: [
                    { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
                    { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                    { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                    endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },

```

```

        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
}
},
{
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
        taskID: 22,
        taskName: 'Implementation Module 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'Critical',
        approved: false,
        duration: 12,
        progress: 90,
        subtasks: [
            { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
            { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
            { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
                endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
                endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
            { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),

```

```

        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  }],
},
{
  taskID: 29,
  taskName: 'Phase 3',
  startDate: new Date('02/17/2017'),
  endDate: new Date('02/27/2017'),
  priority: 'Normal',
  approved: false,
  duration: 11,
  progress: 30,
  subtasks: [{
    taskID: 30,
    taskName: 'Implementation Module 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'High',
    approved: false,
    duration: 11,
    progress: 60,
    subtasks: [
      { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
      { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
      { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
      { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
      { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
      { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
  }]
}
]
}
];

```

MAIN.TS

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

2. Using the properties in the FilterEventArgs object

Alternatively, you can use the properties available in the [FilterEventArgs](#) object to obtain the filter record details.

- [columns](#): This property returns the collection of filtered columns.
- [currentFilterObject](#): This property returns the object that is currently filtered.
- [currentFilteringColumn](#): This property returns the column name that is currently filtered.

To access these properties, you can use the [actionComplete](#) event handler as shown below:

```
`typescript
actionComplete(args: FilterEventArgs) {
var column = args.columns;
var object = args.currentFilterObject;
var name = args.currentFilteringColumn;
}
```

Clear filtering using methods

The Tree Grid provides a method called [clearFiltering](#) to clear the filtering applied to the tree grid. This method is used to remove the filter conditions and reset the tree grid to its original state.

Here's an example of how to clear the filtering in a tree grid using the [clearFiltering](#) method:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { FilterSettingsModel, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridAllModule, ButtonModule
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: ` <button ej2-button cssClass="e-primary"
(click)="onClick()">Clear filter</button>
```



```

        <ejs-treegrid #treegrid id='treegrid' [dataSource]='data'
[treeColumnIndex]='1' height='275' [allowFiltering]='true'
childMapping='subtasks' >
            <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
                <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
                <e-column field='startDate' headerText='Start Date'
textAlign='Right' type='date' format='yMd' width=90></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
            </e-columns>
        </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        @ViewChild('treegrid')
        public treegrid:TreeGridComponent | undefined;
        public data?: Object[];

        ngOnInit(): void {
            this.data = sampleData;
        }
        public onClick(): void {
            this.treegrid?.clearFiltering(); //clear filtering for all columns
        }
    }

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },

```

```

        { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
          endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
      ]
    },
    {
      taskID: 6,
      taskName: 'Design',
      startDate: new Date('02/10/2017'),
      endDate: new Date('02/14/2017'),
      duration: 3,
      progress: 86,
      priority: 'High',
      approved: false,
      subtasks: [
        { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
          endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
        { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
          endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
          endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
      ]
    },
    {
      taskID: 12,
      taskName: 'Implementation Phase',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'Normal',
      approved: false,
      duration: 11,
      progress: 66,
      subtasks: [
        {
          taskID: 13,
          taskName: 'Phase 1',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/27/2017'),
          priority: 'High',
          approved: false,
          progress: 50,
          duration: 11,

```

```

        subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
                { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
                { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
                { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
            ]
        }
    ],
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [

```

```

        { taskID: 23, taskName: 'Development Task 1',
  startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
  progress: '50', priority: 'Normal', approved: true },
        { taskID: 24, taskName: 'Development Task 2',
  startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
  progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
  Date('02/21/2017'),
            endDate: new Date('02/24/2017'), duration: 2,
  progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
  Date('02/25/2017'),
            endDate: new Date('02/26/2017'), duration: 2,
  progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
  startDate: new Date('02/27/2017'),
            endDate: new Date('02/28/2017'), duration: 2,
  progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
  startDate: new Date('02/28/2017'),
            endDate: new Date('02/28/2017'), duration: 0,
  progress: '50', priority: 'Normal', approved: false }
    ]
  },
  {
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
      taskID: 30,
      taskName: 'Implementation Module 3',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'High',
      approved: false,
      duration: 11,
      progress: 60,
      subtasks: [
        { taskID: 31, taskName: 'Development Task 1',
  startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
  progress: '50', priority: 'Low', approved: true },
        { taskID: 32, taskName: 'Development Task 2',
  startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
  progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
  Date('02/20/2017'),

```

```

        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
  }
}
]
];

```

MAIN.TS

```

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

```

Filtering events

Filtering events allow you to customize the behavior of the tree grid when filtering is applied. You can prevent filtering for specific columns, show messages, or perform other actions to suit your application's needs.

To implement filtering events in the Tree Grid, you can utilize the available events such as [actionBegin](#) and [actionComplete](#). These events allow you to intervene in the filtering process and customize it as needed.

In the given example, the filtering is prevented for **duration** column during **actionBegin** event.

APP.COMPONENT.TS

```

{% raw %}
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit,ViewChild} from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,

```

```

selector: 'app-container',
template: `<div id='message' style='color:red'>{{message}}</div>
<ejs-treegrid #treegrid id='treegrid' [dataSource]='data'
[treeColumnIndex]='1' height='275' [allowFiltering]='true'
childMapping='subtasks' (actionBegin)="actionBegin($event)"
(actionComplete)="actionComplete($event)">
<e-columns>
<e-column field='taskID' headerText='Task ID' textAlign='Right'
width=90></e-column>
<e-column field='taskName' headerText='Task Name' textAlign='Left'
width=180></e-column>
<e-column field='startDate' headerText='Start Date' textAlign='Right'
type='date' format='yMd' width=90></e-column>
<e-column field='duration' headerText='Duration' textAlign='Right'
width=80></e-column>
</e-columns>
</ejs-treegrid>`
  })
  export class AppComponent implements OnInit {
    @ViewChild('treegrid')
    public treegrid: TreeGridComponent | undefined;
    public data?: Object[];
    public message: string = '';
    ngOnInit(): void {
      this.data = sampleData;
    }
    actionBegin(args: any) {
      if (args.requestType == 'filtering' && args.currentFilteringColumn ==
        'duration') {
        args.cancel = true;
        this.message = 'The ' + args.type + ' event has been triggered and the ' +
          args.requestType + ' action is cancelled for ' + args.currentFilteringColumn
          + ' column.';
      }
    }
    actionComplete(args: any) {
      if (args.requestType == 'filtering' && args.currentFilteringColumn) {
        this.message = 'The ' + args.type + ' event has been triggered and the ' +
          args.requestType + ' action for the ' + args.currentFilteringColumn + '
          column has been successfully executed';
      } else {
        this.message = '';
      }
    }
  }
  {% endraw %}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',

```

```

        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
                priority: 'Low', approved: true }
        ],
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
                priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
                priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
                priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
                priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
                endDate: new Date('02/14/2017'), duration: 0, progress: 0,
                priority: 'Normal', approved: true }
        ],
    },
],

```

```

{
  taskID: 12,
  taskName: 'Implementation Phase',
  startDate: new Date('02/17/2017'),
  endDate: new Date('02/27/2017'),
  priority: 'Normal',
  approved: false,
  duration: 11,
  progress: 66,
  subtasks: [
    {
      taskID: 13,
      taskName: 'Phase 1',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'High',
      approved: false,
      progress: 50,
      duration: 11,
      subtasks: [{
        taskID: 14,
        taskName: 'Implementation Module 1',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        duration: 11,
        progress: 10,
        approved: false,
        subtasks: [
          { taskID: 15, taskName: 'Development Task 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
            progress: '50', priority: 'High', approved: false },
          { taskID: 16, taskName: 'Development Task 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
            progress: '50', priority: 'Low', approved: true },
          { taskID: 17, taskName: 'Testing', startDate: new
            Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
            progress: '0', priority: 'Normal', approved: true },
          { taskID: 18, taskName: 'Bug fix', startDate: new
            Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
            progress: '0', priority: 'Critical', approved: false },
          { taskID: 19, taskName: 'Customer review meeting',
            startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
            progress: '0', priority: 'High', approved: false },
          { taskID: 20, taskName: 'Phase 1 complete',
            startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
            progress: '50', priority: 'Low', approved: true }
        ]
      }
    ]
  },
  {

```



```

        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
                progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
                progress: '50', priority: 'Critical', approved: true },
                { taskID: 25, taskName: 'Testing', startDate: new
                Date('02/21/2017'),
                endDate: new Date('02/24/2017'), duration: 2,
                progress: '0', priority: 'High', approved: false },
                { taskID: 26, taskName: 'Bug fix', startDate: new
                Date('02/25/2017'),
                endDate: new Date('02/26/2017'), duration: 2,
                progress: '0', priority: 'Low', approved: false },
                { taskID: 27, taskName: 'Customer review meeting',
                startDate: new Date('02/27/2017'),
                endDate: new Date('02/28/2017'), duration: 2,
                progress: '0', priority: 'Critical', approved: true },
                { taskID: 28, taskName: 'Phase 2 complete',
                startDate: new Date('02/28/2017'),
                endDate: new Date('02/28/2017'), duration: 0,
                progress: '50', priority: 'Normal', approved: false }
            ]
        }
    ]
},
{
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
        taskID: 30,
        taskName: 'Implementation Module 3',

```

```

        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'High',
        approved: false,
        duration: 11,
        progress: 60,
        subtasks: [
            { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
            { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
        ]
    }
}
];

```

MAIN.TS

```

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

```

Filter bar in Angular TreeGrid component

The filter bar feature provides a user-friendly way to filter data in the Syncfusion Angular Tree Grid. It displays an input field for each column, allowing you to enter filter criteria and instantly see the filtered results.

By defining the [allowFiltering](#) to true, then filter bar row will be rendered next to header which allows you to filter data. You can filter the records with different expressions depending upon the column type.

Filter bar expressions:

You can enter the following filter expressions(operators) manually in the filter bar.

Expression | Example | Description | Column Type

= | =value | equal | Number

!= | !=value | notequal | Number

|>value | greaterthan | Number

< | <value | lessthan | Number

= | >=value | greaterthanorequal | Number

<= | <=value | lessthanorequal | Number

- | *value | startswith | String

% | %value | endswith | String

N/A | N/A | Always **equal** operator will be used for Date filter | Date

N/A | N/A | Always **equal** operator will be used for Boolean filter | Boolean

The following example demonstrates how to activate default filtering in the tree grid.

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from '../datasource';
import { FilterSettingsModel } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' height='275'
[treeColumnIndex]='1' [allowFiltering]='true'
[filterSettings]='filterSettings' childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' type='date' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
```

```

public filterSettings?: FilterSettingsModel;
ngOnInit(): void {
    this.data = sampleData;
    this.filterSettings = {
        type: 'FilterBar'
    }
}
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ],
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),

```

```

        endDate: new Date('02/12/2017'), duration: 3, progress: 60,
        priority: 'Normal', approved: false },
        { taskID: 8, taskName: 'Develop prototype', startDate: new
        Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
        priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
        new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
        Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
        Date('02/14/2017'),
        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
        priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            progress: 50,
            duration: 11,
            subtasks: [{
                taskID: 14,
                taskName: 'Implementation Module 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'Normal',
                duration: 11,
                progress: 10,
                approved: false,
                subtasks: [
                    { taskID: 15, taskName: 'Development Task 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'High', approved: false },
                    { taskID: 16, taskName: 'Development Task 2',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'Low', approved: true },

```

```

        { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
      taskID: 22,
      taskName: 'Implementation Module 2',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/28/2017'),
      priority: 'Critical',
      approved: false,
      duration: 12,
      progress: 90,
      subtasks: [
        { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
        { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),

```

```

        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
    }]
},
{
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
        taskID: 30,
        taskName: 'Implementation Module 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'High',
        approved: false,
        duration: 11,
        progress: 60,
        subtasks: [
            { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
            { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: 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 enable or dynamically switch the filter type, you must set the [filterSettings.type](#) as **FilterBar**.

Filter bar modes

In the TreeGrid component, the filter bar can operate in two different ways when filtering criteria are applied. These modes, "OnEnter Mode" and "Immediate Mode," provide different experiences and behaviors when interacting with the filter bar. The **OnEnter** mode is the default mode of the filter bar in the TreeGrid component.

OnEnter Mode:

By settings [filterSettings.mode](#) as **OnEnter**, the filter bar captures the filter criteria entered but doesn't initiate filtering until the **Enter** key is pressed. This allows multiple criteria modifications without triggering immediate filtering actions.

Immediate Mode:

By settings [filterSettings.mode](#) as **Immediate**, the filter bar instantly applies filtering as filter criteria are entered. Filtering actions take place as soon as criteria are input or modified, providing real-time previews of filtering results.

The following example illustrates how to enable different filter bar modes in the tree grid.

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { FilterBarMode, FilterSettingsModel } from '@syncfusion/ej2-angular-grids';
import { ChangeEventArgs } from '@syncfusion/ej2-angular-dropdowns';
@Component({
  imports: [
    TreeGridAllModule, DropDownListAllModule
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<div class='input-container'>
    <label for='fields' class='label'>Select Filter
    Mode</label>
    <ejs-dropdownlist #field id='fields'
    [dataSource]='filterModesData' (change)='onModeChange($event)'></ejs-
    dropdownlist>
```



```

        </div>

        <ejs-treegrid [dataSource]='data' height='230'
[treeColumnIndex]='1' [allowFiltering]='true'
[filterSettings]='filterSettings' childMapping='subtasks' >
            <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
                <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
                <e-column field='startDate' headerText='Start Date'
textAlign='Right' type='date' format='yMd' width=90></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
            </e-columns>
        </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public filterSettings?: FilterSettingsModel;
        public filterModesData:any = ['Immediate', 'OnEnter'];
        ngOnInit(): void {
            this.data = sampleData;
            this.filterSettings = {
                type:'FilterBar'
            }
        }
        onModeChange(args: ChangeEventArgs): void {
            this.filterSettings = {
                mode: args.value as FilterBarMode,
            }
        }
    };
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),

```

```

        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
        { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ]
},
{
    taskID: 6,
    taskName: 'Design',
    startDate: new Date('02/10/2017'),
    endDate: new Date('02/14/2017'),
    duration: 3,
    progress: 86,
    priority: 'High',
    approved: false,
    subtasks: [
        { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
        { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',

```

```

        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'High',
        approved: false,
        progress: 50,
        duration: 11,
        subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
                { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
                { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
                { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
            ]
        }]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),

```

```

        priority: 'Critical',
        approved: false,
        duration: 12,
        progress: 90,
        subtasks: [
            { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
            { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
            { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
            endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
            endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
            { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
            endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
        ]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),

```

```

        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
    }]
    }
  ]
}
];

```

MAIN.TS

```

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

```

Display filter text in pager

The TreeGrid component provides an option to display filter text within the pager, indicating the current filtering status. Enabling this feature provides you with a clear understanding of the applied filters and the criteria used for filtering.

To enable the display of filter text within the pager, you should set the [showFilterBarStatus](#) property within the [filterSettings](#) configuration.

The following example demonstrates how to display the filter text in the pager of the tree grid by using the [showFilterBarStatus](#) property:

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService, PageService } from
'@syncfusion/ej2-angular-treegrid';
import { SwitchModule } from '@syncfusion/ej2-angular-buttons';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { FilterBarMode, FilterSettingsModel } from '@syncfusion/ej2-angular-
grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid'

```

```

@Component({
  imports: [
    TreeGridAllModule, SwitchModule
  ],
  providers: [FilterService, PageService],
  standalone: true,
  selector: 'app-container',
  template: `<div class='container'>
    <label for="checked"> <b> Show filter bar status </b>
  </label>
    <ejs-switch id="checked" [checked]="true"
  (change)="onChange($event)"></ejs-switch>
  </div>

    <ejs-treegrid #treegrid [dataSource]='data' height='230'
  [treeColumnIndex]='1' [allowPaging]='true' [allowFiltering]='true'
  childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
    textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
    textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
    textAlign='Right' type='date' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
    textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];

  @ViewChild('treegrid')
  public treegrid: TreeGridComponent | undefined;
  ngOnInit(): void {
    this.data = sampleData;
  }
  onChange(args: any): void {
    (this.treegrid as TreeGridComponent).filterSettings = {
      showFilterBarStatus: args.checked,
    };
  }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date('02/03/2017'),
    endDate: new Date('02/07/2017'),
  }
]

```

```

        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ],
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
                endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
        ],
    },
    {
        taskID: 12,

```

```

    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
      {
        taskID: 13,
        taskName: 'Phase 1',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'High',
        approved: false,
        progress: 50,
        duration: 11,
        subtasks: [{
          taskID: 14,
          taskName: 'Implementation Module 1',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/27/2017'),
          priority: 'Normal',
          duration: 11,
          progress: 10,
          approved: false,
          subtasks: [
            { taskID: 15, taskName: 'Development Task 1',
              startDate: new Date('02/17/2017'),
              endDate: new Date('02/19/2017'), duration: 3,
              progress: '50', priority: 'High', approved: false },
            { taskID: 16, taskName: 'Development Task 2',
              startDate: new Date('02/17/2017'),
              endDate: new Date('02/19/2017'), duration: 3,
              progress: '50', priority: 'Low', approved: true },
            { taskID: 17, taskName: 'Testing', startDate: new
              Date('02/20/2017'),
              endDate: new Date('02/21/2017'), duration: 2,
              progress: '0', priority: 'Normal', approved: true },
            { taskID: 18, taskName: 'Bug fix', startDate: new
              Date('02/24/2017'),
              endDate: new Date('02/25/2017'), duration: 2,
              progress: '0', priority: 'Critical', approved: false },
            { taskID: 19, taskName: 'Customer review meeting',
              startDate: new Date('02/26/2017'),
              endDate: new Date('02/27/2017'), duration: 2,
              progress: '0', priority: 'High', approved: false },
            { taskID: 20, taskName: 'Phase 1 complete',
              startDate: new Date('02/27/2017'),
              endDate: new Date('02/27/2017'), duration: 0,
              progress: '50', priority: 'Low', approved: true }
          ]
        }
      ]
    },
    {
      taskID: 21,
      taskName: 'Phase 2',

```



```

        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
                { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
                endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
                endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
                { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
                { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
                endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
            ]
        }
    ]
},
{
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
        taskID: 30,
        taskName: 'Implementation Module 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),

```

```

        priority: 'High',
        approved: false,
        duration: 11,
        progress: 60,
        subtasks: [
            { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
            { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
        ]
    }
}
];

```

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 filter bar operator in filter bar cell

In the TreeGrid component, you have the ability to modify the filter operator for a column directly within the interface while filtering through the filter bar cell. For instance, the default operator for filtering string-type columns in the filter bar is "startswith". Now, you can customize the default operator for a specific column using the filter operator feature.

To achieve this functionality, you can enable the [showFilterBarOperator](#) property within the [filterSettings](#) of the grid object using the tree grid instance in the [load](#) event of the tree grid.

The following example demonstrates how to show or hide the filter bar operator in the filter bar cell of the tree grid by using the [showFilterBarOperator](#) property.

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { SwitchModule } from '@syncfusion/ej2-angular-buttons';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from '../datasource';
import { FilterBarMode, FilterSettingsModel } from '@syncfusion/ej2-angular-grids';
import { TreeGridComponent, FilterService, PageService } from '@syncfusion/ej2-angular-treegrid'
@Component({
  imports: [
    TreeGridAllModule, SwitchModule
  ],
  providers: [FilterService, PageService],
  standalone: true,
  selector: 'app-container',
  template: `<div class='container'>
    <label for="checked"> <b> Show filter bar operator </b> </label>
    <ejs-switch id="checked" [checked]=true
    (change)="onChange($event)"></ejs-switch>
  </div>

  <ejs-treegrid #treegrid [dataSource]='data' height='230'
  [treeColumnIndex]='1' [allowPaging]='true' [allowFiltering]='true'
  childMapping='subtasks' (load)=load($event) >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
      textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
      textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
      textAlign='Right' type='date' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
      textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];

  @ViewChild('treegrid')
  public treegrid: TreeGridComponent | undefined;
  ngOnInit(): void {
    this.data = sampleData;
  }
  load(args: any): void {
    (this.treegrid as TreeGridComponent).grid.filterSettings = {
      showFilterBarOperator: true,
    };
  }
  onChange(args: any): void {
    (this.treegrid as TreeGridComponent).grid.filterSettings = {

```

```

        showFilterBarOperator: args.checked,
    };
}
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ],
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),

```

```

        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            progress: 50,
            duration: 11,
            subtasks: [{
                taskID: 14,
                taskName: 'Implementation Module 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'Normal',
                duration: 11,
                progress: 10,
                approved: false,
                subtasks: [
                    { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
                    { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                    { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                    endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },

```

```

        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
}
},
{
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
        taskID: 22,
        taskName: 'Implementation Module 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'Critical',
        approved: false,
        duration: 12,
        progress: 90,
        subtasks: [
            { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
            { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
            { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
            endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
            endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
            { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),

```

```

        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  }],
},
{
  taskID: 29,
  taskName: 'Phase 3',
  startDate: new Date('02/17/2017'),
  endDate: new Date('02/27/2017'),
  priority: 'Normal',
  approved: false,
  duration: 11,
  progress: 30,
  subtasks: [{
    taskID: 30,
    taskName: 'Implementation Module 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'High',
    approved: false,
    duration: 11,
    progress: 60,
    subtasks: [
      { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
      { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
      { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
      { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
      { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
      { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
  }]
}
]
}
];

```

MAIN.TS

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

Hide filter bar for template column

By default, the filter bar is set to a disabled mode for template columns in the grid. However, in certain cases, you may want to hide the filter bar for a template column to provide a customized filtering experience.

To hide the filter bar for a template column, you can use the [filterTemplate](#) property of the [column](#) and remove the html content of the filter bar. This property allows you to define a custom template for the filter bar of a column.

Here's an example that demonstrates how to hide the filter bar for a template column in the tree grid:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService, PageService } from
 '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { FilterBarMode, FilterSettingsModel } from '@syncfusion/ej2-angular-
grids';
import { Column, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid'
import { isNullOrUndefined } from '@syncfusion/ej2-base';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService, PageService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data' height='230'
[treeColumnIndex]='1' [allowPaging]='true' [allowFiltering]='true'
(load)="load()" childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column headerText="Action" width="150">
        <ng-template #template let-data>
          <button ejs-button >Custom action</button>
        </ng-template>
      </e-column></e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];

  @ViewChild('treegrid')
  public treegrid: TreeGridComponent | undefined;
  ngOnInit(): void {
```



```

        this.data = sampleData;
    }
    load() {
        //Here find the index of template column
        var column:any=(this.treegrid as TreeGridComponent).columns;
        var
templatecol_inx:any=column.findIndex((x:any)=>!isNullOrUndefined(x.template)
);

        // Set filterTemplate to an empty span to hide the filter bar for the
template column
        ((this.treegrid as TreeGridComponent).columns[templatecol_inx] as
Column).filterTemplate = '<span></span>';
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),

```

```

        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
              endDate: new Date('02/12/2017'), duration: 3, progress: 60,
              priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
              endDate: new Date('02/12/2017'), duration: 3, progress: 100,
              priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
              endDate: new Date('02/14/2017'), duration: 2, progress: 100,
              priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
              endDate: new Date('02/14/2017'), duration: 2, progress: 100,
              priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
              endDate: new Date('02/14/2017'), duration: 0, progress: 0,
              priority: 'Normal', approved: true }
        ]
    },
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,
                    taskName: 'Implementation Module 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'Normal',
                    duration: 11,
                    progress: 10,
                    approved: false,
                    subtasks: [

```

```

        { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
        { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
        { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
          endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
          endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
          endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
          endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
      ]
    },
  ],
  {
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
      taskID: 22,
      taskName: 'Implementation Module 2',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/28/2017'),
      priority: 'Critical',
      approved: false,
      duration: 12,
      progress: 90,
      subtasks: [
        { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
        { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),

```

```

        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
                { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
                { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            ]
        }
    ]
    }
    ]
}

```

```

        { taskID: 36, taskName: 'Phase 3 complete',
  startDate: new Date('02/27/2017'),
              endDate: new Date('02/27/2017'), duration: 0,
  progress: '50', priority: 'Critical', approved: false },
      ]
    }
  ]
}
];

```

MAIN.TS

```

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

```

Filter bar template with custom component

Normally, text box is the default element rendered in the filter bar cell. You can customize the components displayed in the filter bar cell using the [filterBarTemplateLink to the Video](#) feature. This flexibility allows you to use various components, such as datepicker, numerictextbox, combobox, and multiselect, within the filter bar based on your specific requirements.

You can check this video to learn about how to use custom component in filter bar in Angular Tree Grid.

To utilize this feature, you can define a custom template for the filter bar by setting the `filterBarTemplate` property of a column in your Angular application.

The following example demonstrates how to show a custom component in the filter bar cell of the tree grid by using the `filterBarTemplate` property.

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule, FilterService, PageService } from
 '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleGridData } from './datasource';
import { IFilterUI, parentsUntil } from '@syncfusion/ej2-angular-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import {
  ComboBox,
  DropDownList,
  MultiSelect,
} from '@syncfusion/ej2-angular-dropdowns';
import { DataManager, DataUtil, Predicate, Query } from '@syncfusion/ej2-
data';
import { DatePicker } from '@syncfusion/ej2-angular-calendars';
import { NumericTextBox } from '@syncfusion/ej2-angular-inputs';
@Component({
  imports: [
    TreeGridAllModule,
  ],

```

```

    providers: [FilterService, PageService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid #treegrid [dataSource]='data' height='275'
[treeColumnIndex]='1' [allowFiltering]='true' idMapping='taskID'
parentIdMapping= "parentID" >
        <e-columns>
            <e-column field="taskID" headerText="Task ID"
width="120" textAlign="Right" isPrimaryKey="true"></e-column>
            <e-column field="taskName" headerText="Task Name"
width="120" [filterBarTemplate]="templateOptionsDropDown"></e-column>
            <e-column field="duration" headerText="Duration"
width="100" format="C2"
[filterBarTemplate]="templateOptionsNumericTextBox"></e-column>
            <e-column field="startDate" headerText="Start Date"
width="120" format="yMd" type="date"
[filterBarTemplate]="templateOptionsDatePicker"></e-column>
            <e-column field="description" headerText="Task
Description" width="120" [filterBarTemplate]="templateOptionsComboBox"></e-
column>
            <e-column field="category" headerText="Category"
width="120" [filterBarTemplate]="templateOptionsMultiSelect"></e-column>
        </e-columns>
    </ejs-treegrid>`
  })
  export class AppComponent implements OnInit {
    @ViewChild('treegrid')
    public treegrid?: TreeGridComponent;
    public data?: Object[];
    public templateOptionsDropDown?: IFilterUI;
    public templateOptionsNumericTextBox?: IFilterUI;
    public templateOptionsDatePicker?: IFilterUI;
    public templateOptionsComboBox?: IFilterUI;
    public templateOptionsMultiSelect?: IFilterUI;
    public categoryDistinctData?: object[];
    public descriptionDistinctData?: object[];

    public dropdown?: HTMLElement;
    public numElement?: HTMLInputElement;
    public dateElement?: HTMLInputElement;
    public comboelement?: HTMLElement;
    public multiselectelement?: HTMLElement;
    public handleFilterChange(args: { element: Element; value: string }) {
      let targetElement = parentsUntil(args.element, 'e-filtertext');
      let columnName: string = targetElement.id.replace('_filterBarcell',
      '');
      if (args.value) {
        (this.treegrid as TreeGridComponent).filterByColumn(columnName,
        'equal', args.value);
      } else {
        (this.treegrid as
TreeGridComponent).removeFilteredColsByField(columnName);
      }
    }
    public multiselectFunction(args: { value: string }) {
      var selectedValues = args.value;
      if (selectedValues.length === 0) {

```

```

        var OriginalData = new DataManager(this.data).executeLocal(new
Query());
        (this.treegrid as TreeGridComponent).dataSource = OriginalData;
    } else {
        var predicate: Predicate | null = null;
        for (let x = 0; x < selectedValues.length; x++) {
            if (predicate === null) {
                predicate = new Predicate('category', 'equal',
selectedValues[x]);
            } else {
                predicate = predicate.or('category', 'equal',
selectedValues[x]);
            }
        }
        var filteredData = new DataManager(this.data).executeLocal(new
Query().where(predicate as Predicate));
        (this.treegrid as TreeGridComponent).dataSource = filteredData;
    }
}

public dropdownFunction(args: { value: string; item: { parentElement: {
id: string } } }) {
    if (args.value !== 'All') {
        (this.treegrid as
TreeGridComponent).filterByColumn(args.item.parentElement.id.replace('_optio
ns', ''), 'equal', args.value);
    } else {
        (this.treegrid as
TreeGridComponent).removeFilteredColsByField(args.item.parentElement.id.repl
ace('_options', ''));
    }
}

public ngOnInit(): void {
    this.data = sampleGridData;

    this.descriptionDistinctData = DataUtil.distinct(this.data,
'description', true);
    this.categoryDistinctData = DataUtil.distinct(this.data, 'category',
true);
    this.templateOptionsDropDown = {
        create: () => {
            this.dropdown = document.createElement('select');
            this.dropdown.id = 'taskName';
            var option = document.createElement('option');
            option.value = 'All';
            option.innerText = 'All';
            this.dropdown.appendChild(option);
            (this.data as Object[]).forEach((item: object) => {
                const option = document.createElement('option');
                option.value = (item as ItemType).taskName;
                option.innerText = (item as ItemType).taskName;
                (this.dropdown as HTMLElement).appendChild(option);
            });
            return this.dropdown;
        },
        write: () => {
            const dropdownlist = new DropDownList({

```

```

        change: this.dropdownFunction.bind(this),
    });
    dropdownlist.appendTo(this.dropdown);
},
};
this.templateOptionsNumericTextBox = {
    create: () => {
        this.numElement = document.createElement('input');
        return this.numElement;
    },
    write: () => {
        const numericTextBox = new NumericTextBox({
            format: '00.00',
            value: 10,
        });
        numericTextBox.appendTo(this.numElement);
    },
};
this.templateOptionsDatePicker = {
    create: () => {
        this.dateElement = document.createElement('input');
        return this.dateElement;
    },
    write: (args: { column: { field: string | number | Date } }) =>
    {
        const datePickerObj = new DatePicker({
            value: new Date(args.column.field),
            change: this.handleFilterChange.bind(this),
        });
        datePickerObj.appendTo(this.dateElement);
    },
};
this.templateOptionsComboBox = {
    create: () => {
        this.comboelement = document.createElement('input');
        this.comboelement.id = 'description';
        return this.comboelement;
    },
    write: (args: { value: string }) => {
        const comboBox = new ComboBox({
            value: args.value,
            placeholder: 'Select a city',
            change: this.handleFilterChange.bind(this),
            dataSource: (this.descriptionDistinctData as
object[]).map(
                (item: object) => (item as ItemType).description
            ),
        });
        comboBox.appendTo(this.comboelement);
    },
};
this.templateOptionsMultiSelect = {
    create: () => {
        this.multiselectelement = document.createElement('input');
        this.multiselectelement.id = 'category';
        return this.multiselectelement;
    },
};

```



```

        write: (args: { value: string[] | number[] | boolean[] |
undefined }) => {
            const multiselect = new MultiSelect({
                value: args.value,
                placeholder: 'Select a country',
                change: this.multiselectFunction.bind(this),
                dataSource: (this.categoryDistinctData as object[]).map(
                    (item: object) => (item as ItemType).category
                ),
            });
            multiselect.appendTo(this.multiselectelement);
        },
    };
}

interface ItemType {
    taskName: string,
    description: string,
    category: string
}

```

DATASOURCE.TS

```

export let sampleGridData: Object[] = [
    { taskID: 1, taskName: 'Planning', startDate: new
Date('02/03/2023'), endDate: new Date('02/07/2023'), progress: 100, duration:
5,
        priority: 'Normal', approved: false, description: 'Task description
1', category: 'Task category 1', parentID: null },
    { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2023'), description: 'Child task description 1', parentID: 1,
        category: 'Child task category 1', endDate: new Date('02/07/2023'),
duration: 5, progress: 100, priority: 'Normal', approved: false },
    { taskID: 3, taskName: 'Plan budget', startDate: new Date('02/03/2023'),
description: 'Child task description 2', parentID: 1,
        category: 'Child task category 2', endDate: new Date('02/07/2023'),
duration: 5, progress: 100, priority: 'Low', approved: true },
    { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2023'), description: 'Child task description 3', parentID: 1,
        category: 'Child task category 3', endDate: new Date('02/07/2023'),
duration: 5, progress: 100, priority: 'Critical', approved: false },
    { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2023'), description: 'Child task description 4', parentID: 1,
        category: 'Child task category 4', endDate: new Date('02/07/2023'),
duration: 0, progress: 0, priority: 'Low', approved: true },
    { taskID: 6, taskName: 'Design', startDate: new Date('02/10/2023'),
endDate: new Date('02/14/2023'), duration: 3, progress: 86,
        priority: 'High', approved: false, description: 'Task description
2', category: 'Task category 2', parentID: null },
    { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2023'), description: 'Child task description 5', parentID: 6,
        category: 'Child task category 5', endDate: new Date('02/12/2023'),
duration: 3, progress: 60, priority: 'Normal', approved: false },
    { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2023'), description: 'Child task description 6', parentID: 6,

```

```

        category: 'Child task category 6', endDate: new Date('02/12/2023'),
        duration: 3, progress: 100, priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate: new
        Date('02/13/2023'), description: 'Child task description 7',parentID:6,
        category: 'Child task category 7', endDate: new Date('02/14/2023'),
        duration: 2, progress: 100, priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
        Date('02/13/2023'), description: 'Child task description 8',parentID:6,
        category: 'Child task category 8',endDate: new Date('02/14/2023'),
        duration: 2, progress: 100, priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
        Date('02/14/2023'), description: 'Child task description 9',parentID:6,
        category: 'Child task category 9', endDate: new Date('02/14/2023'),
        duration: 0, progress: 0, priority: 'Normal', approved: true },
        { taskID: 12, taskName: 'Implementation Phase',startDate: new
        Date('02/17/2023'),endDate: new Date('02/27/2023'), priority: 'Normal',
        approved: false,description: 'Task description 3', category: 'Task
        category 3', duration: 11,progress: 66, parentID:null},
        { taskID: 13,taskName: 'Phase 1',startDate: new
        Date('02/17/2023'),endDate: new Date('02/27/2023'),priority:
        'High',approved: false,
        progress: 50, description: 'Child task description 10', category:
        'Child task category 10',parentID:12,duration: 11},
        { tasID: 21, taskName: 'Phase 2', startDate: new Date('02/17/2023'),
        endDate: new Date('02/28/2023'),priority: 'High',approved: false,
        duration: 12,description: 'Child task description 11',category:
        'Child task category 11',parentID:12,progress: 60},
        { taskID: 29, taskName: 'Phase 3', startDate: new Date('02/17/2023'),
        endDate: new Date('02/27/2023'), priority: 'Normal',approved: false,
        duration: 11,description: 'Child task description 12', category:
        'Child task category 12',parentID:12, progress: 30}
    ];

```

MAIN.TS

```

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

```

Filter menu in Angular TreeGrid component

The filter menu in the Angular TreeGrid component allows you to enable filtering and provides a user-friendly interface for filtering data based on column types and operators.

To enable the filter menu, you need to set the [filterSettings.type](#) property to **Menu**. This property determines the type of filter UI that will be rendered. The filter menu UI allows you to apply filters using different operators.

Here is an example that demonstrates the usage of the filter menu in the tree grid:

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';

```

```

import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='275' [allowFiltering]='true' [filterSettings]="filterSettings"
childMapping='subtasks' >
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=150></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' type='date' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=120></e-column>
      </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public filterSettings?: Object;
  ngOnInit(): void {
    this.data = sampleData;
    this.filterSettings = {type: 'Menu'};
  }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date('02/03/2017'),
    endDate: new Date('02/07/2017'),
    progress: 100,
    duration: 5,
    priority: 'Normal',
    approved: false,
    subtasks: [
      { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
      { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),

```

```

        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
        { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ]
},
{
    taskID: 6,
    taskName: 'Design',
    startDate: new Date('02/10/2017'),
    endDate: new Date('02/14/2017'),
    duration: 3,
    progress: 86,
    priority: 'High',
    approved: false,
    subtasks: [
        { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
        { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',

```

```

        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'High',
        approved: false,
        progress: 50,
        duration: 11,
        subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
                { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
                { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
                { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
            ]
        }
    ]
},
{
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
        taskID: 22,
        taskName: 'Implementation Module 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),

```

```

        priority: 'Critical',
        approved: false,
        duration: 12,
        progress: 90,
        subtasks: [
            { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
            { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
            { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
            endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
            endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
            { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
            endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
        ]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),

```

```

        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
  }
}
]
};

```

MAIN.TS

```

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

```

* [allowFiltering](#) must be set as true to enable filter menu.

* By setting [columns.allowFiltering](#) as false will prevent filter menu rendering for a particular column.

Custom component in filter menu

You can enhance the filtering experience in the TreeGrid component by customizing the filter menu with custom components. This allows you to replace the default search box with custom components like dropdowns or textboxes. By default, the filter menu provides an autocomplete component for string type columns, a numeric textbox for number type columns, and a dropdown component for boolean type columns, making it easy to search for values.

To customize the filter menu, you can make use of the [column.filter.ui](#) property. This property allows you to integrate your desired custom filter component into a specific column of the tree grid. To implement a custom filter UI, you need to define the following functions:

- **create:** This function is responsible for creating the custom component for the filter.
- **write:** The write function is used to wire events for the custom component. This allows you to handle changes in the custom filter UI.
- **read:** The read function is responsible for reading the filter value from the custom component. This is used to retrieve the selected filter value.

For example, you can replace the standard search box in the filter menu with a dropdown component. This enables you to perform filtering operations by selecting values from the dropdown list, rather than manually typing in search queries.

Here is a sample code demonstrating how to render a dropdownlist component for the **Task Name** column:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData, dropdowndata } from './datasource';
import { DataManager } from '@syncfusion/ej2-data';
import { DropDownList } from '@syncfusion/ej2-angular-dropdowns';
import { createElement } from '@syncfusion/ej2-base';
import { FilterSettingsModel, IFilter, Filter, Column } from '@syncfusion/ej2-angular-grids';

@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='275' [allowFiltering]='true' [filterSettings]="filterSettings"
childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
[filter]='filter' textAlign='Left' width=150></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' type='date' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=120></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public filterSettings?: Object;
  public filter?: IFilter;
  public dropInstance?: DropDownList;
  ngOnInit(): void {
    this.data = sampleData;
    this.filterSettings = {type: 'Menu'};
    this.filter = {
      ui: {
        create: (args: { target: Element, column: object }) => {
          const flValInput: HTMLElement = createElement('input', {
            className: 'flm-input' });
          args.target.appendChild(flValInput);
          this.dropInstance = new DropDownList({
```



```

        dataSource: new DataManager(dropdowndata),
        fields: { text: 'taskName', value: 'taskName' },
        placeholder: 'Select a value',
        popupHeight: '200px'
    });
    (this.dropInstance as
DropDownList).appendTo(flValInput);
    },
    write: (args: {
        column: object, target: Element,
        filteredValue: string
    }) => {
        (this.dropInstance as DropDownList).value =
    (args).filteredValue as string;
    },
    read: (args: { target: Element, column: Column, operator:
string, fltrObj: Filter }) => {
        args.fltrObj.filterByColumn(args.column.field,
args.operator, ((this.dropInstance as DropDownList).value as string));
    }
    }
    };
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let dropdowndata = ['Planning', 'Plan timeline', 'Plan budget', 'Allocate
resources', 'Planning complete',
'Design', 'Software Specification', 'Develop prototype', 'Get approval from
customer', 'Design Documentation', 'Design complete',
'Implementation Phase', 'Phase 1', 'Implementation Module 1', 'Development Task
1', 'Development Task 2', 'Testing',
'Bug fix', 'Customer review meeting', 'Phase 1 complete', 'Phase
2', 'Implementation Module 2', 'Phase 2 complete',
'Phase 3', 'Implementation Module 3', 'Phase 3 complete',
];
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },

```

```

        { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
          endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
        { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
          endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
          endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
      ],
    },
    {
      taskID: 6,
      taskName: 'Design',
      startDate: new Date('02/10/2017'),
      endDate: new Date('02/14/2017'),
      duration: 3,
      progress: 86,
      priority: 'High',
      approved: false,
      subtasks: [
        { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
          endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
        { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
          endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
          endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
      ]
    },
    {
      taskID: 12,
      taskName: 'Implementation Phase',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'Normal',
      approved: false,
      duration: 11,
      progress: 66,
      subtasks: [
        {

```

```

        taskID: 13,
        taskName: 'Phase 1',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'High',
        approved: false,
        progress: 50,
        duration: 11,
        subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
                { taskID: 15, taskName: 'Development Task 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
                progress: '50', priority: 'High', approved: false },
                { taskID: 16, taskName: 'Development Task 2',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
                progress: '50', priority: 'Low', approved: true },
                { taskID: 17, taskName: 'Testing', startDate: new
                Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
                progress: '0', priority: 'Normal', approved: true },
                { taskID: 18, taskName: 'Bug fix', startDate: new
                Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
                progress: '0', priority: 'Critical', approved: false },
                { taskID: 19, taskName: 'Customer review meeting',
                startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
                progress: '0', priority: 'High', approved: false },
                { taskID: 20, taskName: 'Phase 1 complete',
                startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
                progress: '50', priority: 'Low', approved: true }
            ]
        }
    ],
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',

```

```

        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'Critical',
        approved: false,
        duration: 12,
        progress: 90,
        subtasks: [
            { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
            { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
            { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
            endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
            endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
            { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
            endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
        ]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },

```

```

        { taskID: 32, taskName: 'Development Task 2',
        startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
        progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
        Date('02/20/2017'),
          endDate: new Date('02/21/2017'), duration: 2,
        progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
        Date('02/24/2017'),
          endDate: new Date('02/25/2017'), duration: 2,
        progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
        startDate: new Date('02/26/2017'),
          endDate: new Date('02/27/2017'), duration: 2,
        progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
        startDate: new Date('02/27/2017'),
          endDate: new Date('02/27/2017'), duration: 0,
        progress: '50', priority: 'Critical', approved: false },
      ]
    }
  ]
}
];

```

MAIN.TS

```

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

```

Show 24 hours time format in filter dialog

The Tree Grid provides a feature to display the time in a 24-hour format in the date or datetime column filter dialog. By default, the filter dialog displays the time in a 12-hour format (AM/PM) for the date or datetime column. To enable the 24-hour time format in the filter dialog, you need to handle the [actionComplete](#) event with [requestType](#) as [filterafteropen](#) and set the [timeFormat](#) of the [DateTimePicker](#) to **HH:mm**.

In Tree Grid column, you can customize the default format by setting the [type](#) as **dateTime** and the [format](#) as **M/d/y HH:mm**.

Here is an example that demonstrates how to show 24 hours time format in filter dialog:

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData ,dropdowndata} from './datasource';
import { DataManager } from '@syncfusion/ej2-data';

```

```

import { DropDownList } from '@syncfusion/ej2-angular-dropdowns';
import { createElement } from '@syncfusion/ej2-base';
import { FilterSettingsModel, IFilter, Filter, Column } from '@syncfusion/ej2-angular-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' height='275' [allowFiltering]='true'
[filterSettings]="filterSettings" (actionComplete)="actionComplete($event)"
childMapping='subtasks' >
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=150></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' type="datetime" [format]="formatoptions" width=150></e-column>
        <e-column field='endDate' headerText='End Date'
textAlign='Right' type="datetime" [format]="formatoptions" width=150></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=120></e-column>
      </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public filterSettings?: Object;
  public formatoptions?: Object;
  @ViewChild('treegrid')
  public treegrid?:TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
    this.filterSettings = {type: 'Menu'};
    this.formatoptions = { type: 'dateTime', format: 'M/d/y HH:mm' };
  }
  public actionComplete(args: { requestType: string; columnName: string }): void {
    if (args.requestType === 'filterafteropen') {
      var columnObj :any=
this.treegrid?.getColumnByField(args.columnName);
      if (columnObj.type === 'datetime') {
        var dateObj = (document.getElementById('dateui-' +
columnObj.uid) as any)['ej2_instances'][0];
        dateObj.timeFormat = 'HH:mm';
      }
    }
  }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date(1696937e6),
        endDate: new Date(1598763e6),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date(1536944e6),
                endDate: new Date(1536951e6), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date(1537951e6),
                endDate: new Date(1538951e6), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date(1536958e6),
                endDate: new Date(1535565e6), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date(1538751e6),
                endDate: new Date(1537781e6), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ],
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date(1638958e6),
        endDate: new Date(1636679e6),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date(1634965e6),
                endDate: new Date(1636586e6), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date(1637972e6),
                endDate: new Date(1638993e6), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date(1633979e6),
        ]
    }
]

```

```

        endDate: new Date(1637800e6), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date(1636986e6),
        endDate: new Date(1637027e6), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date(1636393e6),
        endDate: new Date(1637314e6), duration: 0, progress: 0,
priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date(1637000e6),
    endDate: new Date(1637021e6),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',
            startDate: new Date(1637607e6),
            endDate: new Date(16370548e6),
            priority: 'High',
            approved: false,
            progress: 50,
            duration: 11,
            subtasks: [{
                taskID: 14,
                taskName: 'Implementation Module 1',
                startDate: new Date(1637714e6),
                endDate: new Date(16370235e6),
                priority: 'Normal',
                duration: 11,
                progress: 10,
                approved: false,
                subtasks: [
                    { taskID: 15, taskName: 'Development Task 1',
startDate: new Date(1737021e6),
                    endDate: new Date(1737042e6), duration: 3,
progress: '50', priority: 'High', approved: false },
                    { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2023'),
                    endDate: new Date(1737635e6), duration: 3,
progress: '50', priority: 'Low', approved: true },
                    { taskID: 17, taskName: 'Testing', startDate: new
Date(1737028e6),
                    endDate: new Date(17372328e6), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                    { taskID: 18, taskName: 'Bug fix', startDate: new
Date(1737035e6),
                    endDate: new Date(1737621e6), duration: 2,
progress: '0', priority: 'Critical', approved: false },

```



```

        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date(1737042e6),
          endDate: new Date(1735014e6), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date(1737035e6),
          endDate: new Date(1737057e6), duration: 0,
progress: '50', priority: 'Low', approved: true }
      ]
    },
    {
      taskID: 21,
      taskName: 'Phase 2',
      startDate: new Date(1737028e6),
      endDate: new Date(1737050e6),
      priority: 'High',
      approved: false,
      duration: 12,
      progress: 60,
      subtasks: [{
        taskID: 22,
        taskName: 'Implementation Module 2',
        startDate: new Date(1735021e6),
        endDate: new Date(1736993e6),
        priority: 'Critical',
        approved: false,
        duration: 12,
        progress: 90,
        subtasks: [
          { taskID: 23, taskName: 'Development Task 1',
startDate: new Date(1737014e6),
            endDate: new Date(1737007e6), duration: 4,
progress: '50', priority: 'Normal', approved: true },
          { taskID: 24, taskName: 'Development Task 2',
startDate: new Date(1736007e6),
            endDate: new Date(1735014e6), duration: 4,
progress: '50', priority: 'Critical', approved: true },
          { taskID: 25, taskName: 'Testing', startDate: new
Date(1737000e6),
            endDate: new Date(1734021e6), duration: 2,
progress: '0', priority: 'High', approved: false },
          { taskID: 26, taskName: 'Bug fix', startDate: new
Date(1737993e6),
            endDate: new Date(1733028e6), duration: 2,
progress: '0', priority: 'Low', approved: false },
          { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date(1736886e6),
            endDate: new Date(1736986e6), duration: 2,
progress: '0', priority: 'Critical', approved: true },
          { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date(1735979e6),
            endDate: new Date(1733979e6), duration: 0,
progress: '50', priority: 'Normal', approved: false }
        ]
      }
    ]
  },
}

```

```

        {
            taskID: 29,
            taskName: 'Phase 3',
            startDate: new Date(1736971e6),
            endDate: new Date(1736972e6),
            priority: 'Normal',
            approved: false,
            duration: 11,
            progress: 30,
            subtasks: [{
                taskID: 30,
                taskName: 'Implementation Module 3',
                startDate: new Date(1736965e6),
                endDate: new Date(1736965e6),
                priority: 'High',
                approved: false,
                duration: 11,
                progress: 60,
                subtasks: [
                    { taskID: 31, taskName: 'Development Task 1',
                        startDate: new Date(1736957e6),
                        endDate: new Date(1736958e6), duration: 3,
                        progress: '50', priority: 'Low', approved: true },
                    { taskID: 32, taskName: 'Development Task 2',
                        startDate: new Date(1736951e6),
                        endDate: new Date(1736951e6), duration: 3,
                        progress: '50', priority: 'Normal', approved: false },
                    { taskID: 33, taskName: 'Testing', startDate: new
                        Date(1736943e6),
                        endDate: new Date(1736944e6), duration: 2,
                        progress: '0', priority: 'Critical', approved: true },
                    { taskID: 34, taskName: 'Bug fix', startDate: new
                        Date(1736937e6),
                        endDate: new Date(1736937e6), duration: 2,
                        progress: '0', priority: 'High', approved: false },
                    { taskID: 35, taskName: 'Customer review meeting',
                        startDate: new Date(1736944e6),
                        endDate: new Date(1736943e6), duration: 2,
                        progress: '0', priority: 'Normal', approved: true },
                    { taskID: 36, taskName: 'Phase 3 complete',
                        startDate: new Date(1736944e6),
                        endDate: new Date(1736951e6), duration: 0,
                        progress: '50', priority: 'Critical', approved: false },
                ]
            }]
        }
    ]
};

```

MAIN.TS

```

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

```

Customizing filter menu operators list

The TreeGrid component enables you to customize the default filter operator list by utilizing the [filterSettings.operators](#) property. This feature allows you to define your own set of operators that will be available in the filter menu. You can customize operators for string, number, date, and boolean data types.

The available options for customization are:

- **stringOperator**- defines customized string operator list.
- **numberOperator** - defines customized number operator list.
- **dateOperator** - defines customized date operator list.
- **booleanOperator** - defines customized boolean operator list.

Here is an example of how to customize the filter operators list in the tree grid:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { FilterSettingsModel } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='275' [allowFiltering]='true' [filterSettings]='filterOptions'
childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=150></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' type='date' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=120></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: object[];
  public filterOptions?: FilterSettingsModel;
  ngOnInit(): void {
    this.data = sampleData;
    this.filterOptions = {
      type: 'Menu',
      operators: {
```

```

        stringOperator: [
            { value: 'startsWith', text: 'Starts With' },
            { value: 'endsWith', text: 'Ends With' },
            { value: 'contains', text: 'Contains' },
            { value: 'equal', text: 'Equal' },
            { value: 'notEqual', text: 'Not Equal' }
        ],
        numberOperator: [
            { value: 'equal', text: 'Equal' },
            { value: 'notEqual', text: 'Not Equal' },
            { value: 'greaterThan', text: 'Greater Than' },
            { value: 'lessThan', text: 'Less Than' }
        ],
        dateOperator: [
            { value: 'equal', text: 'Equal' },
            { value: 'notEqual', text: 'Not Equal' },
            { value: 'greaterThan', text: 'After' },
            { value: 'lessThan', text: 'Before' }
        ],
        booleanOperator: [
            { value: 'equal', text: 'Equal' },
            { value: 'notEqual', text: 'Not Equal' }
        ]
    }
};
}

```

DATASOURCE.TS

```

export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),

```

```

        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ],
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
                endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
        ]
    },
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,

```

```

        taskName: 'Implementation Module 1',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        duration: 11,
        progress: 10,
        approved: false,
        subtasks: [
            { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
            { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
            { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),

```

```

        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
        { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  },
  {
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
      taskID: 30,
      taskName: 'Implementation Module 3',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'High',
      approved: false,
      duration: 11,
      progress: 60,
      subtasks: [
        { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
        { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
          endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },

```

```

        { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
  }
}
];

```

MAIN.TS

```

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

```

Filter by multiple keywords using filter menu

The TreeGrid component allows you to perform filtering actions based on multiple keywords, rather than a single keyword, using the filter menu dialog. To enable this feature, you can set [filterSettings.type](#) as **Menu** and render the **MultiSelect** component as a custom component in the filter menu dialog.

Here is an example that demonstrates how to perform filtering by multiple keywords using the filter menu in the tree grid:

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { FilterSettingsModel, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { DataUtil } from '@syncfusion/ej2-data';
import {
  MultiSelect,
} from '@syncfusion/ej2-angular-dropdowns';
import { createElement } from '@syncfusion/ej2-base';
import {
  IFilter,
  Filter,
  Column,
  PredicateModel,
} from '@syncfusion/ej2-angular-grids';

```



```

@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' height='275' [allowFiltering]='true'
[filterSettings]='filterOptions' childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
[filter]="filter" textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
[filter]="filter" textAlign='Left' width=150></e-column>
      <e-column field='priority' headerText='Priority'
[filter]="filter" width=120></e-column>
      <e-column field='duration' headerText='Duration'
[filter]="filter" textAlign='Right' width=120></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: object[];
  public filterOptions?: FilterSettingsModel;
  public filter?: IFilter;
  public dropInstance?: MultiSelect;
  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
    this.filterOptions = {
      type: 'Menu',
    };
    this.filter = {
      ui: {
        create: (args: { target: Element; column: object }) => {
          const flValInput: HTMLElement = createElement('input', {
            className: 'flm-input',
          });
          args.target.appendChild(flValInput);
          const fieldName: string = (args.column as Column).field;
          const dropdownData: string[] =
DataUtil.distinct(this.treegrid?.flatData as any, fieldName) as string[];
          this.dropInstance = new MultiSelect({
            dataSource: dropdownData,
            placeholder: 'Select a value',
            popupHeight: '200px',
            allowFiltering: true,
            mode: 'Delimiter',
          });
          this.dropInstance.appendTo(flValInput);
        },
        write: (args: { column: Column }) => {
          const fieldName: string = (args.column.field);
          const filteredValue: string[] = [];

```

```

        ((this.treegrid as TreeGridComponent).filterSettings.columns
as any).forEach((item: PredicateModel) => {
            if (item.field === fieldName && item.value) {
                filteredValue.push(item.value as string);
            }
        });
        if (filteredValue.length > 0) {
            (this.dropInstance as MultiSelect).value = filteredValue;
        }
    },
    read: (args: {column:Column,operator:string,fltrObj:Filter}) =>
    {
        (this.treegrid as
TreeGridComponent).removeFilteredColsByField(args.column.field);
        args.fltrObj.filterByColumn(
            args.column.field,
            args.operator,
            this.dropInstance?.value as string[]
        );
    },
    },
    },
    };
}
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date(1696937e6),
        endDate: new Date(1598763e6),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date(1536944e6),
                endDate: new Date(1536951e6), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date(1537951e6),
                endDate: new Date(1538951e6), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date(1536958e6),
                endDate: new Date(1535565e6), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date(1538751e6),

```

```

        endDate: new Date(1537781e6), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ],
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date(1638958e6),
        endDate: new Date(1636679e6),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date(1634965e6),
                endDate: new Date(1636586e6), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date(1637972e6),
                endDate: new Date(1638993e6), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date(1633979e6),
                endDate: new Date(1637800e6), duration: 2, progress: 100,
priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date(1636986e6),
                endDate: new Date(1637027e6), duration: 2, progress: 100,
priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date(1636393e6),
                endDate: new Date(1637314e6), duration: 0, progress: 0,
priority: 'Normal', approved: true }
        ]
    },
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date(1637000e6),
        endDate: new Date(1637021e6),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date(1637607e6),
                endDate: new Date(16370548e6),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,

```

```

        taskName: 'Implementation Module 1',
        startDate: new Date(1637714e6),
        endDate: new Date(16370235e6),
        priority: 'Normal',
        duration: 11,
        progress: 10,
        approved: false,
        subtasks: [
            { taskID: 15, taskName: 'Development Task 1',
startDate: new Date(1737021e6),
                endDate: new Date(1737042e6), duration: 3,
progress: '50', priority: 'High', approved: false },
            { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2023'),
                endDate: new Date(1737635e6), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 17, taskName: 'Testing', startDate: new
Date(1737028e6),
                endDate: new Date(17372328e6), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 18, taskName: 'Bug fix', startDate: new
Date(1737035e6),
                endDate: new Date(1737621e6), duration: 2,
progress: '0', priority: 'Critical', approved: false },
            { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date(1737042e6),
                endDate: new Date(1735014e6), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date(1737035e6),
                endDate: new Date(1737057e6), duration: 0,
progress: '50', priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date(1737028e6),
        endDate: new Date(1737050e6),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date(1735021e6),
            endDate: new Date(1736993e6),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date(1737014e6),

```

```

        endDate: new Date(1737007e6), duration: 4,
progress: '50', priority: 'Normal', approved: true },
        { taskID: 24, taskName: 'Development Task 2',
startDate: new Date(1736007e6),
        endDate: new Date(1735014e6), duration: 4,
progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
Date(1737000e6),
        endDate: new Date(1734021e6), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date(1737993e6),
        endDate: new Date(1733028e6), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date(1736886e6),
        endDate: new Date(1736986e6), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date(1735979e6),
        endDate: new Date(1733979e6), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  },
  {
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date(1736971e6),
    endDate: new Date(1736972e6),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
      taskID: 30,
      taskName: 'Implementation Module 3',
      startDate: new Date(1736965e6),
      endDate: new Date(1736965e6),
      priority: 'High',
      approved: false,
      duration: 11,
      progress: 60,
      subtasks: [
        { taskID: 31, taskName: 'Development Task 1',
startDate: new Date(1736957e6),
        endDate: new Date(1736958e6), duration: 3,
progress: '50', priority: 'Low', approved: true },
        { taskID: 32, taskName: 'Development Task 2',
startDate: new Date(1736951e6),
        endDate: new Date(1736951e6), duration: 3,
progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
Date(1736943e6),
        endDate: new Date(1736944e6), duration: 2,
progress: '0', priority: 'Critical', approved: true },

```

```

        { taskID: 34, taskName: 'Bug fix', startDate: new
Date(1736937e6),
                endDate: new Date(1736937e6), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date(1736944e6),
                endDate: new Date(1736943e6), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date(1736944e6),
                endDate: new Date(1736951e6), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
}
]
];

```

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 default input component of filter menu dialog

You have the flexibility to customize the default settings of input components within the menu filter by utilizing the `params` property within the column definition of the [filter](#). This allows you to modify the behavior of specific filter components to better suit your needs.

Column Type	Default component	Customization
API Reference		
String	AutoComplete Eg: { params: { autofill: false } }	AutoComplete API
Number	NumericTextBox Eg: { params: { showSpinButton: false } }	NumericTextBox API
Boolean	DropDownList Eg: { params: { sortOrder:'Ascending'} }	DropDownList API
Date	DatePicker Eg: { params: { weekNumber: true } }	DatePicker API
DateTime	DateTimePicker Eg: { params: { showClearButton: true } }	DateTimePicker API

To know more about the feature, refer to the [Getting Started documentation](#) and [API Reference](#)

In the example provided below, the **Task ID** and **Duration** columns are numeric columns. When you open the filter dialog for these columns, you will notice that a `NumericTextBox` with a spin button is displayed to change or set the filter value. However, using the `params` property, you can hide the spin button specifically for the **Task ID** column.

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { FilterSettingsModel, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' height='275' [allowFiltering]='true'
[filterSettings]='filterOptions' childMapping='subtasks' >
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
[filter]='filterParams' textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=150></e-column>
        <e-column field='priority' headerText='Priority'
width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=120></e-column>
    </e-columns>
</ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: object[];
  public filterOptions?: FilterSettingsModel;
  public filterParams?: object;
  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
    this.filterOptions = {
      type: 'Menu',
    };
    this.filterParams = { params: { showSpinButton: false } };
  }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
  {
    taskID: 1,

```

```

        taskName: 'Planning',
        startDate: new Date(1696937e6),
        endDate: new Date(1598763e6),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date(1536944e6),
                endDate: new Date(1536951e6), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date(1537951e6),
                endDate: new Date(1538951e6), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date(1536958e6),
                endDate: new Date(1535565e6), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date(1538751e6),
                endDate: new Date(1537781e6), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ],
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date(1638958e6),
        endDate: new Date(1636679e6),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date(1634965e6),
                endDate: new Date(1636586e6), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date(1637972e6),
                endDate: new Date(1638993e6), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date(1633979e6),
                endDate: new Date(1637800e6), duration: 2, progress: 100,
priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date(1636986e6),
                endDate: new Date(1637027e6), duration: 2, progress: 100,
priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date(1636393e6),
                endDate: new Date(1637314e6), duration: 0, progress: 0,
priority: 'Normal', approved: true }
        ]
    }
]

```



```

    },
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date(1637000e6),
        endDate: new Date(1637021e6),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date(1637607e6),
                endDate: new Date(16370548e6),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,
                    taskName: 'Implementation Module 1',
                    startDate: new Date(1637714e6),
                    endDate: new Date(16370235e6),
                    priority: 'Normal',
                    duration: 11,
                    progress: 10,
                    approved: false,
                    subtasks: [
                        { taskID: 15, taskName: 'Development Task 1',
                            startDate: new Date(1737021e6),
                            endDate: new Date(1737042e6), duration: 3,
                            progress: '50', priority: 'High', approved: false },
                        { taskID: 16, taskName: 'Development Task 2',
                            startDate: new Date('02/17/2023'),
                            endDate: new Date(1737635e6), duration: 3,
                            progress: '50', priority: 'Low', approved: true },
                        { taskID: 17, taskName: 'Testing', startDate: new
                            Date(1737028e6),
                            endDate: new Date(17372328e6), duration: 2,
                            progress: '0', priority: 'Normal', approved: true },
                        { taskID: 18, taskName: 'Bug fix', startDate: new
                            Date(1737035e6),
                            endDate: new Date(1737621e6), duration: 2,
                            progress: '0', priority: 'Critical', approved: false },
                        { taskID: 19, taskName: 'Customer review meeting',
                            startDate: new Date(1737042e6),
                            endDate: new Date(1735014e6), duration: 2,
                            progress: '0', priority: 'High', approved: false },
                        { taskID: 20, taskName: 'Phase 1 complete',
                            startDate: new Date(1737035e6),
                            endDate: new Date(1737057e6), duration: 0,
                            progress: '50', priority: 'Low', approved: true }
                    ]
                }
            ]
        }
    },

```

```

    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date(1737028e6),
        endDate: new Date(1737050e6),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date(1735021e6),
            endDate: new Date(1736993e6),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
                    startDate: new Date(1737014e6),
                    endDate: new Date(1737007e6), duration: 4,
                    progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
                    startDate: new Date(1736007e6),
                    endDate: new Date(1735014e6), duration: 4,
                    progress: '50', priority: 'Critical', approved: true },
                { taskID: 25, taskName: 'Testing', startDate: new
                    Date(1737000e6),
                    endDate: new Date(1734021e6), duration: 2,
                    progress: '0', priority: 'High', approved: false },
                { taskID: 26, taskName: 'Bug fix', startDate: new
                    Date(1737993e6),
                    endDate: new Date(1733028e6), duration: 2,
                    progress: '0', priority: 'Low', approved: false },
                { taskID: 27, taskName: 'Customer review meeting',
                    startDate: new Date(1736886e6),
                    endDate: new Date(1736986e6), duration: 2,
                    progress: '0', priority: 'Critical', approved: true },
                { taskID: 28, taskName: 'Phase 2 complete',
                    startDate: new Date(1735979e6),
                    endDate: new Date(1733979e6), duration: 0,
                    progress: '50', priority: 'Normal', approved: false }
            ]
        }
    ],
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date(1736971e6),
        endDate: new Date(1736972e6),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,

```

```

        taskName: 'Implementation Module 3',
        startDate: new Date(1736965e6),
        endDate: new Date(1736965e6),
        priority: 'High',
        approved: false,
        duration: 11,
        progress: 60,
        subtasks: [
            { taskID: 31, taskName: 'Development Task 1',
            startDate: new Date(1736957e6),
            endDate: new Date(1736958e6), duration: 3,
            progress: '50', priority: 'Low', approved: true },
            { taskID: 32, taskName: 'Development Task 2',
            startDate: new Date(1736951e6),
            endDate: new Date(1736951e6), duration: 3,
            progress: '50', priority: 'Normal', approved: false },
            { taskID: 33, taskName: 'Testing', startDate: new
            Date(1736943e6),
            endDate: new Date(1736944e6), duration: 2,
            progress: '0', priority: 'Critical', approved: true },
            { taskID: 34, taskName: 'Bug fix', startDate: new
            Date(1736937e6),
            endDate: new Date(1736937e6), duration: 2,
            progress: '0', priority: 'High', approved: false },
            { taskID: 35, taskName: 'Customer review meeting',
            startDate: new Date(1736944e6),
            endDate: new Date(1736943e6), duration: 2,
            progress: '0', priority: 'Normal', approved: true },
            { taskID: 36, taskName: 'Phase 3 complete',
            startDate: new Date(1736944e6),
            endDate: new Date(1736951e6), duration: 0,
            progress: '50', priority: 'Critical', approved: false },
        ]
    }
}
];

```

MAIN.TS

```

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

```

Prevent autofill option in autocomplete of menu filter

By default, the [AutoComplete](#) component in the filter menu dialog is set to automatically fill suggestions as you type. However, there might be scenarios where you want to prevent this autofill behavior to provide a more customized and controlled user experience.

You can prevent autofill feature by setting the [autofill](#) parameter to **false** using the **params** property within the column definition of the [filter](#).

Here's an example that demonstrates how to prevent autofill options in the autocomplete menu filter within the tree grid. In this example, autofill options have been prevented in the **Task Name** column.

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { FilterSettingsModel, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' height='275' [allowFiltering]='true'
[filterSettings]='filterOptions' childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
[filter]='filterParams' textAlign='Left' width=150></e-column>
      <e-column field='priority' headerText='Priority'
width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=120></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: object[];
  public filterOptions?: FilterSettingsModel;
  public filterParams?: object;
  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
    this.filterOptions = {
      type: 'Menu',
    };
    this.filterParams = { params: { autofill: false } };
  }
}
```

DATASOURCE.TS

```
/**
 * TreeGrid DataSource
 */
```

```

export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date(1696937e6),
    endDate: new Date(1598763e6),
    progress: 100,
    duration: 5,
    priority: 'Normal',
    approved: false,
    subtasks: [
      { taskID: 2, taskName: 'Plan timeline', startDate: new
Date(1536944e6),
        endDate: new Date(1536951e6), duration: 5, progress: 100,
priority: 'Normal', approved: false },
      { taskID: 3, taskName: 'Plan budget', startDate: new
Date(1537951e6),
        endDate: new Date(1538951e6), duration: 5, progress: 100,
priority: 'Low', approved: true },
      { taskID: 4, taskName: 'Allocate resources', startDate: new
Date(1536958e6),
        endDate: new Date(1535565e6), duration: 5, progress: 100,
priority: 'Critical', approved: false },
      { taskID: 5, taskName: 'Planning complete', startDate: new
Date(1538751e6),
        endDate: new Date(1537781e6), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 6,
    taskName: 'Design',
    startDate: new Date(1638958e6),
    endDate: new Date(1636679e6),
    duration: 3,
    progress: 86,
    priority: 'High',
    approved: false,
    subtasks: [
      { taskID: 7, taskName: 'Software Specification', startDate: new
Date(1634965e6),
        endDate: new Date(1636586e6), duration: 3, progress: 60,
priority: 'Normal', approved: false },
      { taskID: 8, taskName: 'Develop prototype', startDate: new
Date(1637972e6),
        endDate: new Date(1638993e6), duration: 3, progress: 100,
priority: 'Critical', approved: false },
      { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date(1633979e6),
        endDate: new Date(1637800e6), duration: 2, progress: 100,
priority: 'Low', approved: true },
      { taskID: 10, taskName: 'Design Documentation', startDate: new
Date(1636986e6),
        endDate: new Date(1637027e6), duration: 2, progress: 100,
priority: 'High', approved: true },
      { taskID: 11, taskName: 'Design complete', startDate: new
Date(1636393e6),

```

```

        endDate: new Date(1637314e6), duration: 0, progress: 0,
priority: 'Normal', approved: true }
    ],
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date(1637000e6),
        endDate: new Date(1637021e6),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date(1637607e6),
                endDate: new Date(16370548e6),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,
                    taskName: 'Implementation Module 1',
                    startDate: new Date(1637714e6),
                    endDate: new Date(16370235e6),
                    priority: 'Normal',
                    duration: 11,
                    progress: 10,
                    approved: false,
                    subtasks: [
                        { taskID: 15, taskName: 'Development Task 1',
startDate: new Date(1737021e6),
                            endDate: new Date(1737042e6), duration: 3,
progress: '50', priority: 'High', approved: false },
                        { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2023'),
                            endDate: new Date(1737635e6), duration: 3,
progress: '50', priority: 'Low', approved: true },
                        { taskID: 17, taskName: 'Testing', startDate: new
Date(1737028e6),
                            endDate:new Date(17372328e6),duration: 2,
progress: '0', priority: 'Normal', approved: true },
                        { taskID: 18, taskName: 'Bug fix', startDate: new
Date(1737035e6),
                            endDate: new Date(1737621e6), duration: 2,
progress: '0', priority: 'Critical', approved: false },
                        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date(1737042e6),
                            endDate: new Date(1735014e6), duration: 2,
progress: '0', priority: 'High', approved: false },
                        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date(1737035e6),
                            endDate: new Date(1737057e6), duration: 0,
progress: '50', priority: 'Low', approved: true }
                    ]
                }
            ]
        }
    ]
}

```

```

    ]
    }],
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date(1737028e6),
        endDate: new Date(1737050e6),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date(1735021e6),
            endDate: new Date(1736993e6),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
                    startDate: new Date(1737014e6),
                    endDate: new Date(1737007e6), duration: 4,
                    progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
                    startDate: new Date(1736007e6),
                    endDate: new Date(1735014e6), duration: 4,
                    progress: '50', priority: 'Critical', approved: true },
                { taskID: 25, taskName: 'Testing', startDate: new
                    Date(1737000e6),
                    endDate: new Date(1734021e6), duration: 2,
                    progress: '0', priority: 'High', approved: false },
                { taskID: 26, taskName: 'Bug fix', startDate: new
                    Date(1737993e6),
                    endDate: new Date(1733028e6), duration: 2,
                    progress: '0', priority: 'Low', approved: false },
                { taskID: 27, taskName: 'Customer review meeting',
                    startDate: new Date(1736886e6),
                    endDate: new Date(1736986e6), duration: 2,
                    progress: '0', priority: 'Critical', approved: true },
                { taskID: 28, taskName: 'Phase 2 complete',
                    startDate: new Date(1735979e6),
                    endDate: new Date(1733979e6), duration: 0,
                    progress: '50', priority: 'Normal', approved: false }
            ]
        }
    ]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date(1736971e6),
        endDate: new Date(1736972e6),
        priority: 'Normal',
        approved: false,
        duration: 11,
    }
]

```

```

        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date(1736965e6),
            endDate: new Date(1736965e6),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
                startDate: new Date(1736957e6),
                endDate: new Date(1736958e6), duration: 3,
                progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
                startDate: new Date(1736951e6),
                endDate: new Date(1736951e6), duration: 3,
                progress: '50', priority: 'Normal', approved: false },
                { taskID: 33, taskName: 'Testing', startDate: new
                Date(1736943e6),
                endDate: new Date(1736944e6), duration: 2,
                progress: '0', priority: 'Critical', approved: true },
                { taskID: 34, taskName: 'Bug fix', startDate: new
                Date(1736937e6),
                endDate: new Date(1736937e6), duration: 2,
                progress: '0', priority: 'High', approved: false },
                { taskID: 35, taskName: 'Customer review meeting',
                startDate: new Date(1736944e6),
                endDate: new Date(1736943e6), duration: 2,
                progress: '0', priority: 'Normal', approved: true },
                { taskID: 36, taskName: 'Phase 3 complete',
                startDate: new Date(1736944e6),
                endDate: new Date(1736951e6), duration: 0,
                progress: '50', priority: 'Critical', approved: false },
            ]
        }]
    }
]
};

```

MAIN.TS

```

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

```

Filter menu events

The Tree Grid offers the [actionBegin](#) and [actionComplete](#) events, which provide information about the actions being performed. Within the event handlers, you receive an argument named `requestType`. This argument specifies the [action](#) that is being executed, such as `filterbeforeopen`, `filterafteropen`, or `filtering`. By analyzing this action type, you can implement custom logic or showcase messages.

filtering - Defines current action as filtering.

filterbeforeopen - Defines current action as filter dialog before open.

filterafteropen - Defines current action as filter dialog after open.

Here's an example of how to use these events to handle filter menu action in the tree grid:

APP.COMPONENT.TS

```
{% raw %}
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { FilterSettingsModel, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<div class='message'
style="color:red">{{actionBeginMessage}}</div><div class='message'
style="color:blue">{{actionCompleteMessage}}</div>
<ejs-treegrid #treegrid [dataSource]='data' [treeColumnIndex]='1'
height='275' [allowFiltering]='true' [filterSettings]='filterOptions'
(actionBegin)="actionBegin($event)"
(actionComplete)="actionComplete($event)" childMapping='subtasks' >
<e-columns>
<e-column field='taskID' headerText='Task ID'      textAlign='Right'
width=90></e-column>
<e-column field='taskName' headerText='Task Name'  textAlign='Left'
width=150></e-column>
<e-column field='priority' headerText='Priority'    width=120></e-column>
<e-column field='duration' headerText='Duration'   textAlign='Right'
width=120></e-column>
</e-columns>
</ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: object[];
  public filterOptions?: FilterSettingsModel;
  public actionBeginMessage: string | undefined;
  public actionCompleteMessage: string | undefined;
  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
    this.filterOptions = {
      type: 'Menu',
    };
  }
  actionBegin(args: any) {
```

```

this.actionBeginMessage='';
if (args.requestType == 'filterbeforeopen' && args.columnType === "number")
{
args.filterModel.customFilterOperators.numberOperator = [
{ value: "equal", text: "Equal" },
{ value: "notequal", text: "Not Equal" }];
this.actionBeginMessage = 'Filter operators for number column were customized
using the filterbeforeopen action in the actionBegin event';
}
else{
this.actionBeginMessage = args.requestType + ' action is triggered in
actionBegin event'
}
if(args.requestType == 'filtering' && args.currentFilteringColumn ==
'priority'){
args.cancel=true;
this.actionBeginMessage = args.requestType + ' is not allowed for Priority
column';
}
}
actionComplete(args: any) {
if(args.requestType === 'filterafteropen') {
this.actionCompleteMessage = 'Applied CSS for filter dialog during
filterafteropen action';
args.filterModel.dlgDiv.querySelector('.e-dlg-content').style.background =
'#eeeeaa';
args.filterModel.dlgDiv.querySelector('.e-footer-content').style.background
= '#30b0ce';
}
if(args.requestType == 'filtering'){
this.actionCompleteMessage = args.requestType + ' action is triggered in
actionComplete event';
}
}
}
}
{% enddraw %}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
{
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date(1696937e6),
    endDate: new Date(1598763e6),
    progress: 100,
    duration: 5,
    priority: 'Normal',
    approved: false,
    subtasks: [
        { taskID: 2, taskName: 'Plan timeline', startDate: new
Date(1536944e6),

```

```

        endDate: new Date(1536951e6), duration: 5, progress: 100,
priority: 'Normal', approved: false },
        { taskID: 3, taskName: 'Plan budget', startDate: new
Date(1537951e6),
        endDate: new Date(1538951e6), duration: 5, progress: 100,
priority: 'Low', approved: true },
        { taskID: 4, taskName: 'Allocate resources', startDate: new
Date(1536958e6),
        endDate: new Date(1535565e6), duration: 5, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 5, taskName: 'Planning complete', startDate: new
Date(1538751e6),
        endDate: new Date(1537781e6), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ]
},
{
    taskID: 6,
    taskName: 'Design',
    startDate: new Date(1638958e6),
    endDate: new Date(1636679e6),
    duration: 3,
    progress: 86,
    priority: 'High',
    approved: false,
    subtasks: [
        { taskID: 7, taskName: 'Software Specification', startDate: new
Date(1634965e6),
        endDate: new Date(1636586e6), duration: 3, progress: 60,
priority: 'Normal', approved: false },
        { taskID: 8, taskName: 'Develop prototype', startDate: new
Date(1637972e6),
        endDate: new Date(1638993e6), duration: 3, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date(1633979e6),
        endDate: new Date(1637800e6), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date(1636986e6),
        endDate: new Date(1637027e6), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date(1636393e6),
        endDate: new Date(1637314e6), duration: 0, progress: 0,
priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date(1637000e6),
    endDate: new Date(1637021e6),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,

```

```

    subtasks: [
      {
        taskID: 13,
        taskName: 'Phase 1',
        startDate: new Date(1637607e6),
        endDate: new Date(16370548e6),
        priority: 'High',
        approved: false,
        progress: 50,
        duration: 11,
        subtasks: [{
          taskID: 14,
          taskName: 'Implementation Module 1',
          startDate: new Date(1637714e6),
          endDate: new Date(16370235e6),
          priority: 'Normal',
          duration: 11,
          progress: 10,
          approved: false,
          subtasks: [
            { taskID: 15, taskName: 'Development Task 1',
              startDate: new Date(1737021e6),
              endDate: new Date(1737042e6), duration: 3,
              progress: '50', priority: 'High', approved: false },
            { taskID: 16, taskName: 'Development Task 2',
              startDate: new Date('02/17/2023'),
              endDate: new Date(1737635e6), duration: 3,
              progress: '50', priority: 'Low', approved: true },
            { taskID: 17, taskName: 'Testing', startDate: new
              Date(1737028e6),
              endDate: new Date(17372328e6), duration: 2,
              progress: '0', priority: 'Normal', approved: true },
            { taskID: 18, taskName: 'Bug fix', startDate: new
              Date(1737035e6),
              endDate: new Date(1737621e6), duration: 2,
              progress: '0', priority: 'Critical', approved: false },
            { taskID: 19, taskName: 'Customer review meeting',
              startDate: new Date(1737042e6),
              endDate: new Date(1735014e6), duration: 2,
              progress: '0', priority: 'High', approved: false },
            { taskID: 20, taskName: 'Phase 1 complete',
              startDate: new Date(1737035e6),
              endDate: new Date(1737057e6), duration: 0,
              progress: '50', priority: 'Low', approved: true }
          ]
        }
      ]
    },
    {
      taskID: 21,
      taskName: 'Phase 2',
      startDate: new Date(1737028e6),
      endDate: new Date(1737050e6),
      priority: 'High',
      approved: false,
      duration: 12,
      progress: 60,
      subtasks: [{

```

```

        taskID: 22,
        taskName: 'Implementation Module 2',
        startDate: new Date(1735021e6),
        endDate: new Date(1736993e6),
        priority: 'Critical',
        approved: false,
        duration: 12,
        progress: 90,
        subtasks: [
            { taskID: 23, taskName: 'Development Task 1',
startDate: new Date(1737014e6),
                endDate: new Date(1737007e6), duration: 4,
progress: '50', priority: 'Normal', approved: true },
            { taskID: 24, taskName: 'Development Task 2',
startDate: new Date(1736007e6),
                endDate: new Date(1735014e6), duration: 4,
progress: '50', priority: 'Critical', approved: true },
            { taskID: 25, taskName: 'Testing', startDate: new
Date(1737000e6),
                endDate: new Date(1734021e6), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 26, taskName: 'Bug fix', startDate: new
Date(1737993e6),
                endDate: new Date(1733028e6), duration: 2,
progress: '0', priority: 'Low', approved: false },
            { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date(1736886e6),
                endDate: new Date(1736986e6), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date(1735979e6),
                endDate: new Date(1733979e6), duration: 0,
progress: '50', priority: 'Normal', approved: false }
        ]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date(1736971e6),
        endDate: new Date(1736972e6),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date(1736965e6),
            endDate: new Date(1736965e6),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
startDate: new Date(1736957e6),

```

```

        endDate: new Date(1736958e6), duration: 3,
progress: '50', priority: 'Low', approved: true },
        { taskID: 32, taskName: 'Development Task 2',
startDate: new Date(1736951e6),
        endDate: new Date(1736951e6), duration: 3,
progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate:new
Date(1736943e6),
        endDate: new Date(1736944e6), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
Date(1736937e6),
        endDate: new Date(1736937e6), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date(1736944e6),
        endDate: new Date(1736943e6), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date(1736944e6),
        endDate: new Date(1736951e6),duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
  }
}
];

```

MAIN.TS

```

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

```

Troubleshoot filter menu operator issue

When using the filter menu, the UI displays operators for all columns based on the data type of the first data it encounters. If the first data is empty or null, it may not work correctly. To overcome this issue, follow these steps to troubleshoot and resolve it:

Explicitly Define Data Type: When defining columns in your Angular TreeGrid component, make sure to explicitly specify the data type for each column. You can do this using the [type](#) property within the columns configuration. For example:

```
`ts
```

```
<ejs-treegrid #treegrid [dataSource]='data' [treeColumnIndex]='1' childMapping='subtasks'>
```

```
<e-columns>
```

```
<e-column field='taskID' headerText='Task ID' type='number' width=120></e-column>
```

```
<e-column field='taskName' headerText='Taskr Name' type='string' width=150></e-column>
```

```
<!-- Define data types for other columns as needed -->
```

```

</e-columns>
</ejs-treegrid>
`

```

Handle Null or Empty Data: If your data source contains null or empty values, make sure that these values are appropriately handled within your data source or by preprocessing your data to ensure consistency.

Check Data Types in Data Source: Ensure that the data types specified in the column definitions match the actual data types in your data source. Mismatched data types can lead to unexpected behavior.

Excel like filter in Angular TreeGrid component

The Syncfusion TreeGrid component offers an Excel-like filter feature, providing a familiar and user-friendly interface for filtering data within the tree grid. This feature simplifies complex filtering operations on specific columns, allowing for quick data location and manipulation, similar to Microsoft Excel. Excel like filtering is especially useful when dealing with large datasets and complex filtering requirements.

Here is an example showcasing how to render the excel like filter within the tree grid:

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  standalone: true,
  providers: [FilterService],
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='273' [allowFiltering]='true' [filterSettings]="filterSettings"
childMapping='subtasks' >
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=150></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' type='date' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=120></e-column>
      </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public filterSettings?: Object;
  ngOnInit(): void {
    this.data = sampleData;
  }
}

```

```

        this.filterSettings = {type: 'Excel'};
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ],
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
        ],
    },
];

```



```

        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
          endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
      ]
    },
    {
      taskID: 12,
      taskName: 'Implementation Phase',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'Normal',
      approved: false,
      duration: 11,
      progress: 66,
      subtasks: [
        {
          taskID: 13,
          taskName: 'Phase 1',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/27/2017'),
          priority: 'High',
          approved: false,
          progress: 50,
          duration: 11,
          subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
              { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
              { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
              { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
              { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),

```

```

        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
  }
},
{
  taskID: 21,
  taskName: 'Phase 2',
  startDate: new Date('02/17/2017'),
  endDate: new Date('02/28/2017'),
  priority: 'High',
  approved: false,
  duration: 12,
  progress: 60,
  subtasks: [{
    taskID: 22,
    taskName: 'Implementation Module 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'Critical',
    approved: false,
    duration: 12,
    progress: 90,
    subtasks: [
      { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
      { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
      { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
      { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
      { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
      { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  }
]

```

```

    ]]
  },
  {
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
      taskID: 30,
      taskName: 'Implementation Module 3',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'High',
      approved: false,
      duration: 11,
      progress: 60,
      subtasks: [
        { taskID: 31, taskName: 'Development Task 1',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
          progress: '50', priority: 'Low', approved: true },
        { taskID: 32, taskName: 'Development Task 2',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
          progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
          Date('02/20/2017'),
          endDate: new Date('02/21/2017'), duration: 2,
          progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
          Date('02/24/2017'),
          endDate: new Date('02/25/2017'), duration: 2,
          progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
          startDate: new Date('02/26/2017'),
          endDate: new Date('02/27/2017'), duration: 2,
          progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
          startDate: new Date('02/27/2017'),
          endDate: new Date('02/27/2017'), duration: 0,
          progress: '50', priority: 'Critical', approved: false },
      ]
    }]
  }
]
};

```

MAIN.TS

```

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

```

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

* The Excel-like filter feature supports various filter conditions, including text-based, number-based, date-based, and boolean-based filters.

* The filter dialog provides additional options, such as sorting filter values, searching for specific values, and clearing applied filters.

Checkbox filtering

The checkbox filtering feature in Tree Grid enables you to filter data based on checkbox selections within a column. This powerful filtering option simplifies the process of narrowing down data, providing a more efficient and user-friendly experience. The check box filter feature is particularly useful when dealing with columns containing categorical data.

Here is an example showcasing how to render the check box filter within the tree grid:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit ,ViewChild} from '@angular/core';
import { sampleData } from './datasource';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='273' [allowFiltering]='true' [filterSettings]="filterSettings"
childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=150></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' type='date' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=120></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public filterSettings?: Object;
  ngOnInit(): void {
    this.data = sampleData;
    this.filterSettings = {type: 'CheckBox'};
  }
}
```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        ]
    }
]

```

```

        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
          endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
      ]
    },
    {
      taskID: 12,
      taskName: 'Implementation Phase',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'Normal',
      approved: false,
      duration: 11,
      progress: 66,
      subtasks: [
        {
          taskID: 13,
          taskName: 'Phase 1',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/27/2017'),
          priority: 'High',
          approved: false,
          progress: 50,
          duration: 11,
          subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
              { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
              { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
              { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
              { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
              { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),

```

```

        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
                { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
                endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
                endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
                { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
                { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
                endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
            ]
        }
    ]
    },
    {
        taskID: 29,

```

```

        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
                progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
                progress: '50', priority: 'Normal', approved: false },
                { taskID: 33, taskName: 'Testing', startDate: new
                Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
                progress: '0', priority: 'Critical', approved: true },
                { taskID: 34, taskName: 'Bug fix', startDate: new
                Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
                progress: '0', priority: 'High', approved: false },
                { taskID: 35, taskName: 'Customer review meeting',
                startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
                progress: '0', priority: 'Normal', approved: true },
                { taskID: 36, taskName: 'Phase 3 complete',
                startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
                progress: '50', priority: 'Critical', approved: false },
            ]
        }
    ]
}
];

```

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 filter choice count

By default, the filter choice count is set to 1000, which means that the filter dialog will display a maximum of 1000 distinct values for each column as a checkbox list data. This default value ensures that the filter operation remains efficient, even with large datasets. Additionally, the filter dialog retrieves and displays distinct data from the first 1000 records bind to the tree grid to optimize performance, while the remaining records are returned as a result of the search option within the filter dialog.

The TreeGrid component allows you to customize the number of distinct data displayed in the checkbox list of the excel/checkbox type filter dialog. This can be useful when you want to customize the default filter choice count values while using large datasets.

However, you have the flexibility to increase or decrease the filter choice count based on your specific requirements. This can be achieved by adjusting the [filterChoiceCount](#) value.

The following example demonstrates how to customize the filter choice count in the checkbox list of the filter dialog. In the [actionBegin](#) event, you need to check if the [requestType](#) is either [filterChoiceRequest](#) or [filterSearchBegin](#), and then you can set the [filterChoiceCount](#) property to the desired value.

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit ,ViewChild} from '@angular/core';
import { sampleData } from './datasource';
import { FilterSearchBeginEventArgs, } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='273' [allowFiltering]='true' [filterSettings]="filterSettings"
(actionBegin)="actionBegin($event)" childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=150></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' type='date' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=120></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public filterSettings?: Object;
  ngOnInit(): void {
    this.data = sampleData;
    this.filterSettings = {type: 'Excel'};
  }
}
```

```

    }
    actionBegin(args: FilterSearchBeginEventArgs) {
        if (args.requestType === "filterchoicerequest" || args.requestType ===
"filtersearchbegin") {
            args.filterChoiceCount = 3000;
        }
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ],
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),

```

```

        endDate: new Date('02/12/2017'), duration: 3, progress: 60,
        priority: 'Normal', approved: false },
        { taskID: 8, taskName: 'Develop prototype', startDate: new
        Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
        priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
        new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
        Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
        Date('02/14/2017'),
        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
        priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            progress: 50,
            duration: 11,
            subtasks: [{
                taskID: 14,
                taskName: 'Implementation Module 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'Normal',
                duration: 11,
                progress: 10,
                approved: false,
                subtasks: [
                    { taskID: 15, taskName: 'Development Task 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'High', approved: false },
                    { taskID: 16, taskName: 'Development Task 2',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'Low', approved: true },
                ]
            }
        ]
    }
]
}

```

```

        { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
      taskID: 22,
      taskName: 'Implementation Module 2',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/28/2017'),
      priority: 'Critical',
      approved: false,
      duration: 12,
      progress: 90,
      subtasks: [
        { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
        { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),

```

```

        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
    }]
},
{
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
        taskID: 30,
        taskName: 'Implementation Module 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'High',
        approved: false,
        duration: 11,
        progress: 60,
        subtasks: [
            { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
            { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
        ]
    }
    ]
}
]
}

```

```
];
```

MAIN.TS

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

The specified filter choice count value determines the display of unique items as a checkbox list in the Excel/checkbox type filter dialog. This can result in a delay in rendering these checkbox items when opening the filter dialog. Therefore, it is advisable to set a restricted filter choice count value.

Show customized text in checkbox list data

The TreeGrid component provides you with the flexibility to customize the text displayed in the Excel/Checkbox filtering options. This allows you to modify the default text and provide more meaningful and contextual labels for the filtering.

To customize the text in the Excel/Checkbox filter, you can define a `filterItemTemplate` and bind it to the desired column. The `filterItemTemplate` property allows you to create custom templates for filter items. You can use any logic and HTML elements within this template to display the desired text or content.

In the example below, you can see how you can customize the text displayed in the filter checkbox list for the **Approved** column. This is achieved by defining a `filterItemTemplate` within the `<e-column>` element for that specific column. Inside the template, you can use Angular's template syntax to conditionally display **Approved** if the data value is true and **Not approved** if the value is false.

APP.COMPONENT.TS

```
{% raw %}
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit ,ViewChild} from '@angular/core';
import { sampleData } from './datasource';
import { FilterSearchBeginEventArgs, } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='273' [allowFiltering]='true' [filterSettings]="filterSettings"
childMapping='subtasks' >
<e-columns>
<e-column field='taskID' headerText='Task ID' textAlign='Right'
width=90></e-column>
<e-column field='taskName' headerText='Task Name' textAlign='Left'
width=150></e-column>
```

```

<e-column field="approved" headerText="Approved" width="120"
displayAsCheckBox="true" [filter]="columnFilterSettings" >
<ng-template #filterItemTemplate let-data>{{data.approved == true ?
"Approved" : "Not approved"}}
</ng-template>
</e-column>
</e-columns>
</ejs-treegrid>`
})
export class AppComponent implements OnInit {
public data?: Object[];
public filterSettings?: Object;
public columnFilterSettings?: Object;
public filterItemTemplate?: string;
ngOnInit(): void {
this.data = sampleData;
this.filterSettings = {type: 'Excel'};
this.columnFilterSettings = {
type: 'CheckBox',
filterItemTemplate: this.filterItemTemplate,
};
}
}
{% enddraw %}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
{
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date('02/03/2017'),
    endDate: new Date('02/07/2017'),
    progress: 100,
    duration: 5,
    priority: 'Normal',
    approved: false,
    subtasks: [
        { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
            endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
        { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
            endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
        { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
            endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),

```

```

        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ],
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
                endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
        ]
    },
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,

```



```

        taskName: 'Implementation Module 1',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        duration: 11,
        progress: 10,
        approved: false,
        subtasks: [
            { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
            { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
            { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),

```

```

    endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
    { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
    endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
    { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
    endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
    { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
    endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
    { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
    endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
    { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
    endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  },
  {
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
      taskID: 30,
      taskName: 'Implementation Module 3',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'High',
      approved: false,
      duration: 11,
      progress: 60,
      subtasks: [
        { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
        { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
          endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },

```

```

        { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
  }
}
];

```

MAIN.TS

```

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

```

[Show template in checkbox list data](#)

The `filterItemTemplate` property in the Tree Grid allows you to customize the appearance of filter items in the tree grid's filter checkbox list for a specific column. This property is useful when you want to provide a custom UI or additional information within the filter checkbox list, such as icons, text, or any HTML elements, alongside the default filter items.

In this example, you can see how to use the `filterItemTemplate` to render icons along with the category names in the filter checkbox list for the **Category Name** column.

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { categoryData } from './datasource';
import { FilterSearchBeginEventArgs, } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='273' [allowFiltering]='true' [filterSettings]="filterSettings"
childMapping='subtasks' >
    <e-columns>

```

```

        <e-column field="ProductID" headerText="ProductID"
width="120" ></e-column>
        <e-column field="CategoryName" headerText="Category Name"
width="150" [filter]="columnFilterSettings">
            <ng-template #filterItemTemplate let-data><span
[ngClass]="categoryIcons[data.CategoryName]"></span> {{data.CategoryName}}
</ng-template>
        </e-column>
        <e-column field="Discontinued" headerText="Discontinued"
width="100" displayAsCheckBox="true" ></e-column>
    </e-columns>
</ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public filterSettings?: Object;
        public columnFilterSettings?: Object;
        public filterItemTemplate?: string;
        categoryIcons: { [key: string]: string } = {
            Beverages: 'fas fa-coffee',
            Condiments: 'fas fa-leaf',
            Confections: 'fas fa-birthday-cake',
            DairyProducts: 'fas fa-ice-cream',
            Grains: 'fas fa-seedling',
            Meat: 'fas fa-drumstick-bite',
            Produce: 'fas fa-carrot',
            Seafood: 'fas fa-fish',
        };

        ngOnInit(): void {
            this.data = categoryData;
            this.filterSettings = {type: 'Excel'};
            this.columnFilterSettings = {

                filterItemTemplate: this.filterItemTemplate,
            };
        }
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export const categoryData: Object[] = [
    {
        CategoryName: 'Beverages',
        ProductID: 1,
        ProductName: 'Chai',
        SupplierID: 1,
        QuantityPerUnit: '10 boxes x 20 bags',
        UnitPrice: 18.0,
        UnitsInStock: 39,
        Discontinued: true,
        subtasks: [

```

```

    {
      CategoryName: 'DairyProducts',
      ProductID: 2,
      ProductName: 'Chang',
      SupplierID: 1,
      QuantityPerUnit: '24 - 12 oz bottles',
      UnitPrice: 19.0,
      UnitsInStock: 17,
      Discontinued: true,
    },

    {
      CategoryName: 'Seafood',
      ProductID: 3,
      ProductName: 'Aniseed Syrup',
      SupplierID: 1,
      QuantityPerUnit: '12 - 550 ml bottles',
      UnitPrice: 10.0,
      UnitsInStock: 13,
      Discontinued: true,
    },

    {
      CategoryName: 'Beverages',
      ProductID: 4,
      ProductName: "Chef Anton's Cajun Seasoning",
      SupplierID: 2,
      QuantityPerUnit: '48 - 6 oz jars',
      UnitPrice: 22.0,
      UnitsInStock: 53,
      Discontinued: true,
    },

    {
      CategoryName: 'Grains',
      ProductID: 5,
      ProductName: "Chef Anton's Gumbo Mix",
      SupplierID: 2,
      QuantityPerUnit: '36 boxes',
      UnitPrice: 21.35,
      UnitsInStock: 0,
      Discontinued: true,
    }, ], },

    {
      CategoryName: 'DairyProducts',
      ProductID: 6,
      ProductName: "Grandma's Boysenberry Spread",
      SupplierID: 3,
      QuantityPerUnit: '12 - 8 oz jars',
      UnitPrice: 25.0,
      UnitsInStock: 120,
      Discontinued: false,
      subtasks: [
        {

```

```

    CategoryName: 'Meat',
    ProductID: 7,
    ProductName: "Uncle Bob's Organic Dried Pears",
    SupplierID: 3,
    QuantityPerUnit: '12 - 1 lb pkgs.',
    UnitPrice: 30.0,
    UnitsInStock: 15,
    Discontinued: false,
  },

  {
    CategoryName: 'DairyProducts',
    ProductID: 8,
    ProductName: 'Northwoods Cranberry Sauce',
    SupplierID: 3,
    QuantityPerUnit: '12 - 12 oz jars',
    UnitPrice: 40.0,
    UnitsInStock: 6,
    Discontinued: false,
  },

  {
    CategoryName: 'Grains',
    ProductID: 9,
    ProductName: 'Mishi Kobe Niku',
    SupplierID: 4,
    QuantityPerUnit: '18 - 500 g pkgs.',
    UnitPrice: 97.0,
    UnitsInStock: 29,
    Discontinued: true,
  },

  {
    CategoryName: 'Meat',
    ProductID: 10,
    ProductName: 'Ikura',
    SupplierID: 4,
    QuantityPerUnit: '12 - 200 ml jars',
    UnitPrice: 31.0,
    UnitsInStock: 31,
    Discontinued: false,
  },
]},

{
  CategoryName: 'Produce',
  ProductID: 11,
  ProductName: 'Queso Cabrales',
  SupplierID: 5,
  QuantityPerUnit: '1 kg pkg.',
  UnitPrice: 21.0,
  UnitsInStock: 22,
  Discontinued: false,
subtasks:[

  {
    CategoryName: 'Seafood',
    ProductID: 12,

```

```
    ProductName: 'Queso Manchego La Pastora',
    SupplierID: 5,
    QuantityPerUnit: '10 - 500 g pkgs.',
    UnitPrice: 38.0,
    UnitsInStock: 86,
    Discontinued: false,
  },
  {
    CategoryName: 'Seafood',
    ProductID: 13,
    ProductName: 'Konbu',
    SupplierID: 6,
    QuantityPerUnit: '2 kg box',
    UnitPrice: 6.0,
    UnitsInStock: 24,
    Discontinued: true,
  },
  {
    CategoryName: 'Grains',
    ProductID: 14,
    ProductName: 'Tofu',
    SupplierID: 6,
    QuantityPerUnit: '40 - 100 g pkgs.',
    UnitPrice: 23.25,
    UnitsInStock: 35,
    Discontinued: true,
  },
  {
    CategoryName: 'Grains',
    ProductID: 15,
    ProductName: 'Genen Shouyu',
    SupplierID: 6,
    QuantityPerUnit: '24 - 250 ml bottles',
    UnitPrice: 15.5,
    UnitsInStock: 39,
    Discontinued: true,
  },
  {
    CategoryName: 'Meat',
    ProductID: 16,
    ProductName: 'Pavlova',
    SupplierID: 7,
    QuantityPerUnit: '32 - 500 g boxes',
    UnitPrice: 17.45,
    UnitsInStock: 29,
    Discontinued: true,
  },
], },
{
  CategoryName: 'Produce',
  ProductID: 17,
  ProductName: 'Alice Mutton',
  SupplierID: 7,
```

```
    QuantityPerUnit: '20 - 1 kg tins',
    UnitPrice: 39.0,
    UnitsInStock: 0,
    Discontinued: true,

    subtasks:[
      {
        CategoryName: 'Produce',
        ProductID: 18,
        ProductName: 'Carnarvon Tigers',
        SupplierID: 7,
        QuantityPerUnit: '16 kg pkg.',
        UnitPrice: 62.5,
        UnitsInStock: 42,
        Discontinued: false,
      },

      {
        CategoryName: 'Beverages',
        ProductID: 19,
        ProductName: 'Teatime Chocolate Biscuits',
        SupplierID: 8,
        QuantityPerUnit: '10 boxes x 12 pieces',
        UnitPrice: 9.2,
        UnitsInStock: 25,
        Discontinued: false,
      },

      {
        CategoryName: 'Grains',
        ProductID: 20,
        ProductName: "Sir Rodney's Marmalade",
        SupplierID: 8,
        QuantityPerUnit: '30 gift boxes',
        UnitPrice: 81.0,
        UnitsInStock: 40,
        Discontinued: false,
      },

      {
        CategoryName: 'Meat',
        ProductID: 21,
        ProductName: "Sir Rodney's Scones",
        SupplierID: 8,
        QuantityPerUnit: '24 pkgs. x 4 pieces',
        UnitPrice: 10.0,
        UnitsInStock: 3,
        Discontinued: false,
      },

      {
        CategoryName: 'Condiments',
        ProductID: 22,
        ProductName: "Gustaf's Knäckebröd",
        SupplierID: 9,
        QuantityPerUnit: '24 - 500 g pkgs.',
        UnitPrice: 21.0,
```



```

    UnitsInStock: 104,
    Discontinued: false,
  },
  {
    CategoryName: 'Seafood',
    ProductID: 23,
    ProductName: 'Tunnbröd',
    SupplierID: 9,
    QuantityPerUnit: '12 - 250 g pkgs.',
    UnitPrice: 9.0,
    UnitsInStock: 61,
    Discontinued: false,
  },
  {
    CategoryName: 'Confections',
    ProductID: 24,
    ProductName: 'Guaraná Fantástica',
    SupplierID: 10,
    QuantityPerUnit: '12 - 355 ml cans',
    UnitPrice: 4.5,
    UnitsInStock: 20,
    Discontinued: true,
  },
  {
    CategoryName: 'Confections',
    ProductID: 25,
    ProductName: 'NuNuCa Nuß-Nougat-Creme',
    SupplierID: 11,
    QuantityPerUnit: '20 - 450 g glasses',
    UnitPrice: 14.0,
    UnitsInStock: 76,
    Discontinued: false,
  },
]
]]
];

```

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 excel filter dialog using CSS

In the Tree Grid, you have the flexibility to enhance the visual presentation of the excel filter dialog. This can be achieved by utilizing CSS styles to modify the dialog's appearance according to the specific needs and aesthetics of your application.

Removing context menu option

The excel filter dialog includes several features such as **context menu**, **search box**, and **checkbox list** that may not be required in some scenarios. You can remove these options using the **className** attribute in the TreeGrid component.

```
`css

.e-treegrid .e-excelfilter .e-contextmenu-wrapper

{

display: none;

}
```

The following example demonstrates how to remove the context menu option in the excel filter dialog using above mentioned CSS

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import { sampleData } from './datasource';
import { FilterSearchBeginEventArgs, } from '@syncfusion/ej2-angular-grids';
@Component({
  encapsulation: ViewEncapsulation.None,
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='273' [allowFiltering]='true' [filterSettings]="filterSettings"
childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=150></e-column>
      <e-column field="approved" headerText="Approved"
width="120" displayAsCheckBox="true"></e-column>
    </e-columns>
  </ejs-treegrid>`,
  styles:[
    `.e-treegrid .e-excelfilter .e-contextmenu-wrapper {
      display: none;
    }`
  ]
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public filterSettings?: Object;

  ngOnInit(): void {
    this.data = sampleData;
  }
}
```

```

        this.filterSettings = {type: 'Excel'};

    }

}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ],
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),

```

```

        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
        priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
        priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            progress: 50,
            duration: 11,
            subtasks: [{
                taskID: 14,
                taskName: 'Implementation Module 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'Normal',
                duration: 11,
                progress: 10,
                approved: false,
                subtasks: [
                    { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
                    { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                    { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                    endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },

```

```

        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
}
},
{
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
        taskID: 22,
        taskName: 'Implementation Module 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'Critical',
        approved: false,
        duration: 12,
        progress: 90,
        subtasks: [
            { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
            { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
            { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
            endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
            endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
            { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),

```

```

        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  }],
},
{
  taskID: 29,
  taskName: 'Phase 3',
  startDate: new Date('02/17/2017'),
  endDate: new Date('02/27/2017'),
  priority: 'Normal',
  approved: false,
  duration: 11,
  progress: 30,
  subtasks: [{
    taskID: 30,
    taskName: 'Implementation Module 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'High',
    approved: false,
    duration: 11,
    progress: 60,
    subtasks: [
      { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
      { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
      { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
      { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
      { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
      { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
  }]}
}
]
}
];

```

MAIN.TS

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

Render checkbox list data in on-demand for excel/checkbox filtering

The Excel/Checkbox filter type of Tree Grid has a restriction where only the first 1000 unique sorted items are accessible to view in the filter dialog checkbox list content by scrolling. This limitation is in place to avoid any rendering delays when opening the filter dialog. However, the searching and filtering processes consider all unique items in that particular column.

The Excel/Checkbox filter in the tree grid provides an option to load large data sets on-demand during scrolling to improve scrolling limitation functionality. This is achieved by setting the [filterSettings.enableInfiniteScrolling](#) property to **true**. This feature proves especially beneficial for managing extensive datasets, enhancing data loading performance in the checkbox list, and allowing interactive checkbox selection with persistence for the selection based on filtering criteria.

The Excel/Checkbox filter retrieves distinct data in ascending order, governed by its [filterSettings.itemsCount](#) property, with a default value of **50**. As the checkbox list data scroller reaches its end, the next dataset is fetched and displayed, with the notable advantage that this process only requests new checkbox list data without redundantly fetching the existing loaded dataset.

Customize the items count for initial rendering

Based on the items count value, the Excel/Checkbox filter gets unique data and displayed in Excel/Checkbox filter content dialog. You can customize the count of on-demand data rendering for Excel/Checkbox filter by adjusting the [filterSettings.itemsCount](#) property. The default value is **50**

```
`ts
```

```
treegrid.filterSettings = { enableInfiniteScrolling = true, itemsCount = 40 };
```

```
,
```

It is recommended to keep the itemsCount below **300**. Higher values will result in unwanted whitespace because of DOM maintenance and performance degradation.

Customize the loading animation effect

A loading effect is presented to signify that loading is in progress when the checkbox list data scroller reaches the end, and there is a delay in receiving the data response from the server. The loading effect during on-demand data retrieval for Excel/Checkbox filter can be customized using the [filterSettings.loadingIndicator](#) property. The default value is **Shimmer**.

```
`ts
```

```
treegrid.filterSettings = { enableInfiniteScrolling = true, loadingIndicator = 'Spinner' };
```

```
,
```

In the provided example, On-Demand Excel filter has been enabled for the tree grid

```
`ts
```

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

```
import { DataManager, Query, WebApiAdaptor, UrlAdaptor } from '@syncfusion/ej2-data';
```

```

@Component({
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1' [query]='query'
    parentIdMapping='ParentItem' hasChildMapping='isParent' idMapping='TaskID' height=265
    allowPaging='true' allowFiltering='true' [pageSettings]='pageSettings' [filterSettings]='filterSettings'>
    <e-columns>
    <e-column field='TaskID' headerText='Task ID' width='90' textAlign='Right'></e-column>
    <e-column field='TaskName' headerText='Task Name' width='170'></e-column>
    <e-column field='StartDate' headerText='Start Date' width='130' format="yMd" textAlign='Right'></e-
    column>
    <e-column field='Duration' headerText='Duration' width='80' textAlign='Right'></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: DataManager;
  public query?: Query;
  public pageSettings: Object | undefined;
  public filterSettings: Object | undefined;
  ngOnInit(): void {
    this.data = new DataManager({
      url: 'https://ej2services.syncfusion.com/production/web-services/api/SelfReferenceData',
      adaptor: new WebApiAdaptor, crossDomain: true
    });
    this.query = new Query().addParams('ej2treegrid', 'true');
    this.pageSettings = { pageCount: 5 };
    this.filterSettings = { type: 'Excel', enableInfiniteScrolling: true };
  }
}
`css

```

[Hide sorting options in filter dialog](#)

The excel filter dialog includes several features such as **context menu**, **search box**, **sorting option** and **checkbox list**. You can hide the sorting options using the **className** attribute in the TreeGrid component.

```
`css
```



```
.e-excel-ascending,
.e-excel-descending,
.e-separator.e-excel-separator {
display: none;
}
`
`
```

The following example demonstrates how to hide the sorting option in the excel filter dialog using above mentioned CSS

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import { sampleData } from './datasource';
import { FilterSearchBeginEventArgs, } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService],
  standalone: true,
  selector: 'app-container',
  encapsulation: ViewEncapsulation.None,
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='273' [allowFiltering]='true' [filterSettings]="filterSettings"
childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=150></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Left' width=90></e-column>
      <e-column field='priority' headerText='Priority'
textAlign='Left' width=90></e-column>
    </e-columns>
  </ejs-treegrid>`,
  styles: [`.e-excel-ascending,
.e-excel-descending,
.e-separator.e-excel-separator {
  display: none;
}`],
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public filterSettings?: Object;

  ngOnInit(): void {
    this.data = sampleData;
    this.filterSettings = {type: 'Excel'};
  }
}
```

```

    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ],
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),

```

```

        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
        priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
        new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
        Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
        Date('02/14/2017'),
        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
        priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            progress: 50,
            duration: 11,
            subtasks: [{
                taskID: 14,
                taskName: 'Implementation Module 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'Normal',
                duration: 11,
                progress: 10,
                approved: false,
                subtasks: [
                    { taskID: 15, taskName: 'Development Task 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'High', approved: false },
                    { taskID: 16, taskName: 'Development Task 2',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'Low', approved: true },
                    { taskID: 17, taskName: 'Testing', startDate: new
                    Date('02/20/2017'),
                    endDate: new Date('02/21/2017'), duration: 2,
                    progress: '0', priority: 'Normal', approved: true },
                ]
            }
        ]
    }
]
}

```

```

        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
}
},
{
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
        taskID: 22,
        taskName: 'Implementation Module 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'Critical',
        approved: false,
        duration: 12,
        progress: 90,
        subtasks: [
            { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
            { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
            { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
                endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
                endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
            { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),

```

```

        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  }],
},
{
  taskID: 29,
  taskName: 'Phase 3',
  startDate: new Date('02/17/2017'),
  endDate: new Date('02/27/2017'),
  priority: 'Normal',
  approved: false,
  duration: 11,
  progress: 30,
  subtasks: [{
    taskID: 30,
    taskName: 'Implementation Module 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'High',
    approved: false,
    duration: 11,
    progress: 60,
    subtasks: [
      { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
      { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
      { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
      { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
      { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
      { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
  }]
}
]
}
];

```

MAIN.TS

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

Searching in Angular TreeGrid component

The Syncfusion Angular TreeGrid component includes a powerful built-in searching feature that facilitates searching for specific data within the tree grid. This feature enables efficient filtering of tree grid records based on user-defined search criteria, making it easier to locate and display relevant information. Whether you have a large dataset or simply need to find specific records quickly, the search feature provides a convenient solution.

To enable the searching feature in the tree grid, set the [allowSearching](#) property to **true**.

To further enhance the search functionality, you can integrate a search text box directly into the tree grid's toolbar. This allows you to enter search criteria conveniently within the tree grid interface. To add the search item to the tree grid's toolbar, use the [toolbar](#) property and add **Search** item.

Here is an example that demonstrates the default searching feature of the tree grid:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule, FilterService, ToolbarService, } from
 '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService, ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='270' [toolbar]='toolbarOption' childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public toolbarOption?: string[];
  ngOnInit(): void {
    this.data = sampleData;
    this.toolbarOption = ['Search'];
  }
}
```

```
}
```

DATASOURCE.TS

```
/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date('02/03/2017'),
    endDate: new Date('02/07/2017'),
    progress: 100,
    duration: 5,
    priority: 'Normal',
    approved: false,
    subtasks: [
      { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
        priority: 'Normal', approved: false },
      { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
        priority: 'Low', approved: true },
      { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
        priority: 'Critical', approved: false },
      { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
        priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 6,
    taskName: 'Design',
    startDate: new Date('02/10/2017'),
    endDate: new Date('02/14/2017'),
    duration: 3,
    progress: 86,
    priority: 'High',
    approved: false,
    subtasks: [
      { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 60,
        priority: 'Normal', approved: false },
      { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
        priority: 'Critical', approved: false },
      { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
```

```

        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
        Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
        Date('02/14/2017'),
        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
        priority: 'Normal', approved: true }
    ],
    },
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,
                    taskName: 'Implementation Module 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'Normal',
                    duration: 11,
                    progress: 10,
                    approved: false,
                    subtasks: [
                        { taskID: 15, taskName: 'Development Task 1',
                        startDate: new Date('02/17/2017'),
                        endDate: new Date('02/19/2017'), duration: 3,
                        progress: '50', priority: 'High', approved: false },
                        { taskID: 16, taskName: 'Development Task 2',
                        startDate: new Date('02/17/2017'),
                        endDate: new Date('02/19/2017'), duration: 3,
                        progress: '50', priority: 'Low', approved: true },
                        { taskID: 17, taskName: 'Testing', startDate: new
                        Date('02/20/2017'),
                        endDate: new Date('02/21/2017'), duration: 2,
                        progress: '0', priority: 'Normal', approved: true },
                        { taskID: 18, taskName: 'Bug fix', startDate: new
                        Date('02/24/2017'),
                        endDate: new Date('02/25/2017'), duration: 2,
                        progress: '0', priority: 'Critical', approved: false },
                    ]
                }
            ]
        }
    ]
}

```



```

        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
          endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
          endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
      ]
    },
    {
      taskID: 21,
      taskName: 'Phase 2',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/28/2017'),
      priority: 'High',
      approved: false,
      duration: 12,
      progress: 60,
      subtasks: [{
        taskID: 22,
        taskName: 'Implementation Module 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'Critical',
        approved: false,
        duration: 12,
        progress: 90,
        subtasks: [
          { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
          { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
          { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
            endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
          { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
            endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
          { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
          { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
            endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
        ]
      }
    ]
  },
},

```

```

        {
            taskID: 29,
            taskName: 'Phase 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            approved: false,
            duration: 11,
            progress: 30,
            subtasks: [{
                taskID: 30,
                taskName: 'Implementation Module 3',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'High',
                approved: false,
                duration: 11,
                progress: 60,
                subtasks: [
                    { taskID: 31, taskName: 'Development Task 1',
                        startDate: new Date('02/17/2017'),
                        endDate: new Date('02/19/2017'), duration: 3,
                        progress: '50', priority: 'Low', approved: true },
                    { taskID: 32, taskName: 'Development Task 2',
                        startDate: new Date('02/17/2017'),
                        endDate: new Date('02/19/2017'), duration: 3,
                        progress: '50', priority: 'Normal', approved: false },
                    { taskID: 33, taskName: 'Testing', startDate: new
                        Date('02/20/2017'),
                        endDate: new Date('02/21/2017'), duration: 2,
                        progress: '0', priority: 'Critical', approved: true },
                    { taskID: 34, taskName: 'Bug fix', startDate: new
                        Date('02/24/2017'),
                        endDate: new Date('02/25/2017'), duration: 2,
                        progress: '0', priority: 'High', approved: false },
                    { taskID: 35, taskName: 'Customer review meeting',
                        startDate: new Date('02/26/2017'),
                        endDate: new Date('02/27/2017'), duration: 2,
                        progress: '0', priority: 'Normal', approved: true },
                    { taskID: 36, taskName: 'Phase 3 complete',
                        startDate: new Date('02/27/2017'),
                        endDate: new Date('02/27/2017'), duration: 0,
                        progress: '50', priority: 'Critical', approved: false },
                ]
            }]
        }
    ]
};

```

MAIN.TS

```

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

```

The clear icon is shown in the Tree Grid search text box when it is focused on search text or after typing the single character in the search text box. A single click of the clear icon clears the text in the search box as well as the search results in the Tree Grid.

Initial search

By default, the search operation is performed on the tree grid data after the tree grid renders. However, there might be scenarios where need to perform a search operation on the tree grid data during the initial rendering of the tree grid. In such cases, you can make use of the initial search feature provided by the tree grid.

To apply search at initial rendering, need to set the following properties in the [searchSettings](#) object.

Property | Description

startswith | Checks whether a value begins with the specified value.

endswith | Checks whether a value ends with the specified value.

contains | Checks whether a value contains with the specified value.

wildcard | Processes one or more search patterns using the “*” symbol, returning values that match the given patterns.

like | Processes a single search pattern using the “%” symbol, retrieving values that match the specified pattern.

equal | Checks whether a value equal to the specified value.

notequal | Checks whether a value not equal to the specified value.

These operators provide flexibility in defining the search behavior and allow you to perform different types of comparisons based on your requirements.

The following example demonstrates how to set the `searchSettings.operator` property based on changing the dropdown value using the [change](#) event of the [DropDownList](#) component.

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService, ToolbarService, } from
 '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ChangeEventArgs } from '@syncfusion/ej2-angular-dropdowns';
import { TreeGridComponent, ToolbarItems, SearchSettingsModel } from
 '@syncfusion/ej2-angular-treegrid';
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns';
@Component({
  imports: [
    TreeGridAllModule, DropDownListAllModule
  ],
  providers: [FilterService, ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<div style="display: flex">
```

```

        <label style="padding: 10px 10px 26px 0">Change the
search operators:</label>
        <ejs-dropdownlist style="margin-top:5px" id="value"
#dropdown index="0" width="100" [dataSource]="ddlData" [fields]='fields'
(change)="valueChange($event)"></ejs-dropdownlist>
    </div>

    <ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' height='230' [toolbar]='toolbarOptions'
childMapping='subtasks' [searchSettings]='searchSettings' >
        <e-columns>
            <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
            <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
            <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
            <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
        </e-columns>
    </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public toolbarOptions?: ToolbarItems[];
        public searchSettings?: SearchSettingsModel
        @ViewChild('treegrid') public treegrid?: TreeGridComponent;
        public fields?: object = { text: 'text', value: 'value' };
        public ddlData?: object[] = [
            { text: 'startswith', value: 'startswith' },
            { text: 'endswith', value: 'endswith' },
            { text: 'wildcard', value: 'wildcard' },
            { text: 'like', value: 'like' },
            { text: 'equal', value: 'equal' },
            { text: 'not equal', value: 'notequal' },
        ];
        ngOnInit(): void {
            this.data = sampleData;
            this.toolbarOptions = ['Search'];
            this.searchSettings = { operator: 'contains' };
        }
        valueChange(args: ChangeEventArgs): void {
            (this.treegrid as TreeGridComponent).searchSettings.operator =
args.value as string;
        }
    }

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
    }
]

```

```

        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
                priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
                priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
                priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
                priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
                priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
                endDate: new Date('02/14/2017'), duration: 0, progress: 0,
                priority: 'Normal', approved: true }
        ]
    },
],

```

```

{
  taskID: 12,
  taskName: 'Implementation Phase',
  startDate: new Date('02/17/2017'),
  endDate: new Date('02/27/2017'),
  priority: 'Normal',
  approved: false,
  duration: 11,
  progress: 66,
  subtasks: [
    {
      taskID: 13,
      taskName: 'Phase 1',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'High',
      approved: false,
      progress: 50,
      duration: 11,
      subtasks: [{
        taskID: 14,
        taskName: 'Implementation Module 1',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        duration: 11,
        progress: 10,
        approved: false,
        subtasks: [
          { taskID: 15, taskName: 'Development Task 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
            progress: '50', priority: 'High', approved: false },
          { taskID: 16, taskName: 'Development Task 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
            progress: '50', priority: 'Low', approved: true },
          { taskID: 17, taskName: 'Testing', startDate: new
            Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
            progress: '0', priority: 'Normal', approved: true },
          { taskID: 18, taskName: 'Bug fix', startDate: new
            Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
            progress: '0', priority: 'Critical', approved: false },
          { taskID: 19, taskName: 'Customer review meeting',
            startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
            progress: '0', priority: 'High', approved: false },
          { taskID: 20, taskName: 'Phase 1 complete',
            startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
            progress: '50', priority: 'Low', approved: true }
        ]
      }
    ]
  },
  {

```

```

        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/20/2017'), duration: 4,
                    progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/20/2017'), duration: 4,
                    progress: '50', priority: 'Critical', approved: true },
                { taskID: 25, taskName: 'Testing', startDate: new
                    Date('02/21/2017'),
                    endDate: new Date('02/24/2017'), duration: 2,
                    progress: '0', priority: 'High', approved: false },
                { taskID: 26, taskName: 'Bug fix', startDate: new
                    Date('02/25/2017'),
                    endDate: new Date('02/26/2017'), duration: 2,
                    progress: '0', priority: 'Low', approved: false },
                { taskID: 27, taskName: 'Customer review meeting',
                    startDate: new Date('02/27/2017'),
                    endDate: new Date('02/28/2017'), duration: 2,
                    progress: '0', priority: 'Critical', approved: true },
                { taskID: 28, taskName: 'Phase 2 complete',
                    startDate: new Date('02/28/2017'),
                    endDate: new Date('02/28/2017'), duration: 0,
                    progress: '50', priority: 'Normal', approved: false }
            ]
        }
    ]
},
{
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
        taskID: 30,
        taskName: 'Implementation Module 3',

```

```

        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'High',
        approved: false,
        duration: 11,
        progress: 60,
        subtasks: [
            { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
            { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
        ]
    }
}
];

```

MAIN.TS

```

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

```

Search by external button

The TreeGrid component allows you to perform searches programmatically, enabling you to search for records using an external button instead of relying solely on the built-in search bar. This feature provides flexibility and allows for custom search implementations within your application. To search for records using an external button, you can utilize the [search](#) method provided by the TreeGrid component.

The `search` method allows you to perform a search operation based on a search key or criteria. The following example demonstrates how to implement `search` by an external button using the following steps:

1. Add a button element outside of the TreeGrid component.
2. Attach a click event handler to the button.
3. Inside the event handler, get the reference of the TreeGrid component.
4. Invoke the `search` method of the tree grid by passing the search key as a parameter.

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule, FilterService, ToolbarService, } from
'@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { TextBoxComponent } from '@syncfusion/ej2-angular-inputs';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { TextBoxModule } from '@syncfusion/ej2-angular-inputs';
@Component({
  imports: [
    TreeGridAllModule, ButtonModule, TextBoxModule
  ],
  providers: [FilterService, ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<div class="e-float-input" style="width: 120px; display:
inline-block;">
      <ejs-textbox #searchInput width="100"
placeholder="Search text"></ejs-textbox>
      <span class="e-float-line"></span>
    </div>
    <button ejs-button id='search'
(click)='search()'>Search</button>
    <ejs-treegrid #treegrid [dataSource]='data' height='270'
[treeColumnIndex]='1' childMapping='subtasks'>
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
      </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  @ViewChild('treegrid') public treegrid?: TreeGridComponent;
  @ViewChild('searchInput') public searchInput?: TextBoxComponent;
  ngOnInit(): void {
```

```
        this.data = sampleData;
    }
    search() {
        const searchText: string = (this.searchInput as
        TextBoxComponent).value;
        (this.treegrid as TreeGridComponent).search(searchText);
    }
}
```

DATASOURCE.TS

```
/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
            Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
            priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
            Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
            priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
            Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
            priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
            Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
            priority: 'Low', approved: true }
        ],
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
            Date('02/10/2017'),
```

```

        endDate: new Date('02/12/2017'), duration: 3, progress: 60,
        priority: 'Normal', approved: false },
        { taskID: 8, taskName: 'Develop prototype', startDate: new
        Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
        priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
        new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
        Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
        Date('02/14/2017'),
        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
        priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            progress: 50,
            duration: 11,
            subtasks: [{
                taskID: 14,
                taskName: 'Implementation Module 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'Normal',
                duration: 11,
                progress: 10,
                approved: false,
                subtasks: [
                    { taskID: 15, taskName: 'Development Task 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'High', approved: false },
                    { taskID: 16, taskName: 'Development Task 2',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'Low', approved: true },

```

```

        { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
      taskID: 22,
      taskName: 'Implementation Module 2',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/28/2017'),
      priority: 'Critical',
      approved: false,
      duration: 12,
      progress: 90,
      subtasks: [
        { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
        { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),

```

```

        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
                { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
                { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
            ]
        }
    ]
}
    ]
}

```

```
];
```

MAIN.TS

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

Search specific columns

By default, the [search](#) functionality searches all visible columns. However, if you want to search only specific columns, you can define the specific column's field names in the [searchSettings.fields](#) property. This allows you to narrow down the search to a targeted set of columns, which is particularly useful when dealing with large datasets or tree grids with numerous columns.

The following example demonstrates how to search specific columns such as **taskName** and **progress** by using the [searchSettings.fields](#) property.

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService, ToolbarService, } from
'@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems, SearchSettingsModel, TreeGridComponent } from
'@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService, ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data' height='270'
[treeColumnIndex]='1' [toolbar]='toolbarOptions' childMapping='subtasks'
[searchSettings]='searchSettings'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='progress'
headerText='Progress'></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: object[];
  public toolbarOptions?: ToolbarItems[];
  public searchSettings?: SearchSettingsModel;
  @ViewChild('treegrid') public treegridObj?: TreeGridComponent;
```

```
ngOnInit(): void {  
    this.data = sampleData;  
    this.searchSettings = { fields: ['taskName', 'progress'] };  
    this.toolbarOptions = ['Search'];  
}  
}
```

DATASOURCE.TS

```
/**  
 * TreeGrid DataSource  
 */  
export let sampleData: Object[] = [  
    {  
        taskID: 1,  
        taskName: 'Planning',  
        startDate: new Date('02/03/2017'),  
        endDate: new Date('02/07/2017'),  
        progress: 100,  
        duration: 5,  
        priority: 'Normal',  
        approved: false,  
        subtasks: [  
            { taskID: 2, taskName: 'Plan timeline', startDate: new  
Date('02/03/2017'),  
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,  
priority: 'Normal', approved: false },  
            { taskID: 3, taskName: 'Plan budget', startDate: new  
Date('02/03/2017'),  
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,  
priority: 'Low', approved: true },  
            { taskID: 4, taskName: 'Allocate resources', startDate: new  
Date('02/03/2017'),  
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,  
priority: 'Critical', approved: false },  
            { taskID: 5, taskName: 'Planning complete', startDate: new  
Date('02/07/2017'),  
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,  
priority: 'Low', approved: true }  
        ],  
    },  
    {  
        taskID: 6,  
        taskName: 'Design',  
        startDate: new Date('02/10/2017'),  
        endDate: new Date('02/14/2017'),  
        duration: 3,  
        progress: 86,  
        priority: 'High',  
        approved: false,  
        subtasks: [  
            { taskID: 7, taskName: 'Software Specification', startDate: new  
Date('02/10/2017'),  
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,  
priority: 'Normal', approved: false },  
        ]  
    }  
]
```

```

        { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
          endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
          endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
      ]
    },
    {
      taskID: 12,
      taskName: 'Implementation Phase',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'Normal',
      approved: false,
      duration: 11,
      progress: 66,
      subtasks: [
        {
          taskID: 13,
          taskName: 'Phase 1',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/27/2017'),
          priority: 'High',
          approved: false,
          progress: 50,
          duration: 11,
          subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
              { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
              { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
              { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),

```



```

        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
  }
},
{
  taskID: 21,
  taskName: 'Phase 2',
  startDate: new Date('02/17/2017'),
  endDate: new Date('02/28/2017'),
  priority: 'High',
  approved: false,
  duration: 12,
  progress: 60,
  subtasks: [{
    taskID: 22,
    taskName: 'Implementation Module 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'Critical',
    approved: false,
    duration: 12,
    progress: 90,
    subtasks: [
      { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
      { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
      { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
      { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
      { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
    ]
  }
]
}
}

```

```

        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
          endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
      ]
    }],
  },
  {
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
      taskID: 30,
      taskName: 'Implementation Module 3',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'High',
      approved: false,
      duration: 11,
      progress: 60,
      subtasks: [
        { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
        { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
          endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
          endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
          endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
          endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
      ]
    }
  ]
}
];

```

MAIN.TS

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

Search on each key stroke

The search on each keystroke feature in the Tree Grid enables you to perform real-time searching of tree grid data as they type in the search text box. This functionality provides a seamless and interactive searching experience, allowing you to see the search results dynamically updating in real time as they enter each keystroke in the search box

To achieve this, you need to bind the **keyup** event to the search input element inside the **created** event of the TreeGrid component.

In the following example, the **created** event is bound to the TreeGrid component, and inside the event handler, the **keyup** event is bound to the search input element. Whenever the **keyup** event is triggered, the current **search** string is obtained from the **search** input element, and the **search** method is invoked on the tree grid instance with the current search string as a parameter. This allows the search results to be displayed in real-time as you type in the search box.

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule, FilterService, ToolbarService, } from
 '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems, TreeGridComponent } from '@syncfusion/ej2-angular-
treegrid';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService, ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data' height='270'
[treeColumnIndex]='1' [toolbar]='toolbarOptions' childMapping='subtasks'
(created)='created($event)'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
```

```

    })
    export class AppComponent implements OnInit {
        public data?: object[];
        public toolbarOptions?: ToolbarItems[];
        @ViewChild('treegrid')
        public treegrid?: TreeGridComponent;
        ngOnInit(): void {
            this.data = sampleData;
            this.toolbarOptions = ['Search'];
        }
        created(args: any): void {
            (document.getElementById((this.treegrid as
            TreeGridComponent).grid.element.id + "_searchbar") as
            HTMLElement).addEventListener('keyup', () => {
                (this.treegrid as TreeGridComponent).search(((event as
            MouseEvent).target as HTMLInputElement).value)
            });
        }
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
            Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
            priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
            Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
            priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
            Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
            priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
            Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
            priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 6,

```

```

        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
                priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
                priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
                priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
                priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
                endDate: new Date('02/14/2017'), duration: 0, progress: 0,
                priority: 'Normal', approved: true }
        ]
    },
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,
                    taskName: 'Implementation Module 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'Normal',
                    duration: 11,
                    progress: 10,

```

```

        approved: false,
        subtasks: [
            { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
            { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
            { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
            ]
        }
    ]
}

```

```

        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  },
  {
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
      taskID: 30,
      taskName: 'Implementation Module 3',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'High',
      approved: false,
      duration: 11,
      progress: 60,
      subtasks: [
        { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
        { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),

```

```

                                endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                                { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
                                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
                                ]
                                }]
                                }
                                ]
                                }
];

```

MAIN.TS

```

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

```

Search on each key stroke approach may affect the performance of the application when dealing with a large number of records.

Perform search based on column formatting

By default, the search operation considers the underlying raw data of each cell for searching. However, in some cases, you may want to search based on the formatted data visible to the users. To search data based on column formatting, you can utilize the `grid.valueFormatterService.fromView` method within the [actionBegin](#) event. This method allows you to retrieve the formatted value of a cell and perform searching on each column using the **OR** predicate.

The following example demonstrates how to implement searching based on column formatting in the Tree Grid. In the `actionBegin` event, retrieve the search value from the `getColumns` method. Iterate through the columns and check whether the column has a format specified. If the column has a format specified, use the `grid.valueFormatterService.fromView` method to get the formatted value of the cell. If the formatted value matches the search value, set the **OR** predicate that includes the current column filter and the new filter based on the formatted value.

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService, ToolbarService, } from
 '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems, TreeGridComponent } from '@syncfusion/ej2-angular-
treegrid';
import { SearchEventArgs, KeyboardEventArgs } from '@syncfusion/ej2-angular-
grids';
import { Query, Predicate } from '@syncfusion/ej2-data';
@Component({
  imports: [
    TreeGridAllModule,
  ],

```



```

        providers: [FilterService, ToolbarService],
        standalone: true,
        selector: 'app-container',
        template: `<ejs-treegrid #treegrid [dataSource]='data' height='270'
[treeColumnIndex]='1' [toolbar]='toolbarOptions' childMapping='subtasks'
(actionBegin)="actionBegin($event)" (keyPressed)="keyPressed($event)">
            <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
                <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
                <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
            </e-columns>
        </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: object[];
        public toolbarOptions?: ToolbarItems[];
        @ViewChild('treegrid')
        public treegrid?: TreeGridComponent;
        ngOnInit(): void {
            this.data = sampleData;
            this.toolbarOptions = ['Search'];
        }
        actionBegin(args:any) {
            if (args.requestType == 'searching') {
                args.cancel = true;
                setTimeout(() => {
                    var columns = (this.treegrid as
TreeGridComponent).grid.getColumns();
                    var predicate = null;
                    for (var i = 0; i < columns.length; i++) {

                        var val = (this.treegrid as
TreeGridComponent).grid.valueFormatterService.fromView(
                            args.searchString as string,
                            (columns[i] as any).getParser(),
                            columns[i].type
                        );
                        if (val) {
                            if (predicate == null) {
                                predicate = new
Predicate(columns[i].field, 'contains', val, true, true);
                            } else {
                                predicate =
predicate.or(columns[i].field, 'contains', val, true, true);
                            }
                        }
                    }
                    (this.treegrid as TreeGridComponent).query = new
Query().where(predicate as Predicate);
                }, 200);
            }
        }
    }

```

```

keyPressed(args: any) {
    if (
        args.key == 'Enter' &&
        args.target instanceof HTMLElement &&
        args.target.closest('.e-search') &&
        (args.target as HTMLInputElement).value == ''
    ) {
        args.cancel = true;
        (this.treegrid as TreeGridComponent).query = new Query();
    }
}
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
                priority: 'Low', approved: true }
        ],
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
    }
]

```

```

        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
              endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
              endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
              endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
              endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
              endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
        ]
    },
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,
                    taskName: 'Implementation Module 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'Normal',
                    duration: 11,
                    progress: 10,
                    approved: false,
                    subtasks: [
                        { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                          endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
                    ]
                }
            ]
            }
        ]
    }
]

```

```

        { taskID: 16, taskName: 'Development Task 2',
        startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
        progress: '50', priority: 'Low', approved: true },
        { taskID: 17, taskName: 'Testing', startDate: new
        Date('02/20/2017'),
          endDate: new Date('02/21/2017'), duration: 2,
        progress: '0', priority: 'Normal', approved: true },
        { taskID: 18, taskName: 'Bug fix', startDate: new
        Date('02/24/2017'),
          endDate: new Date('02/25/2017'), duration: 2,
        progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
        startDate: new Date('02/26/2017'),
          endDate: new Date('02/27/2017'), duration: 2,
        progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
        startDate: new Date('02/27/2017'),
          endDate: new Date('02/27/2017'), duration: 0,
        progress: '50', priority: 'Low', approved: true }
      ]
    },
    {
      taskID: 21,
      taskName: 'Phase 2',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/28/2017'),
      priority: 'High',
      approved: false,
      duration: 12,
      progress: 60,
      subtasks: [{
        taskID: 22,
        taskName: 'Implementation Module 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'Critical',
        approved: false,
        duration: 12,
        progress: 90,
        subtasks: [
          { taskID: 23, taskName: 'Development Task 1',
            startDate: new Date('02/17/2017'),
              endDate: new Date('02/20/2017'), duration: 4,
            progress: '50', priority: 'Normal', approved: true },
          { taskID: 24, taskName: 'Development Task 2',
            startDate: new Date('02/17/2017'),
              endDate: new Date('02/20/2017'), duration: 4,
            progress: '50', priority: 'Critical', approved: true },
          { taskID: 25, taskName: 'Testing', startDate: new
            Date('02/21/2017'),
              endDate: new Date('02/24/2017'), duration: 2,
            progress: '0', priority: 'High', approved: false },
          { taskID: 26, taskName: 'Bug fix', startDate: new
            Date('02/25/2017'),

```

```

        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  }
},
{
  taskID: 29,
  taskName: 'Phase 3',
  startDate: new Date('02/17/2017'),
  endDate: new Date('02/27/2017'),
  priority: 'Normal',
  approved: false,
  duration: 11,
  progress: 30,
  subtasks: [{
    taskID: 30,
    taskName: 'Implementation Module 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'High',
    approved: false,
    duration: 11,
    progress: 60,
    subtasks: [
      { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
      { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
      { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
      { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
      { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
      { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
  }
]
}

```

```

    }
  }
]
];

```

MAIN.TS

```

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

```

Perform search operation in Grid using multiple keywords

In addition to searching with a single keyword, the TreeGrid component offers the capability to perform a search operation using multiple keywords. This feature enables you to narrow down your search results by simultaneously matching multiple keywords. It can be particularly useful when you need to find records that meet multiple search conditions simultaneously. This can be achieved by the [actionBegin](#) event of the Tree Grid.

The following example demonstrates, how to perform a search with multiple keywords in the tree grid by using the [query](#) property when the [requestType](#) is searching in the [actionBegin](#) event. The [searchString](#) is divided into multiple keywords using a comma (,) as the delimiter. Each keyword is then utilized to create a [predicate](#) that checks for a match in the desired columns. If multiple keywords are present, the predicates are combined using an **OR** condition. Finally, the Tree Grid's [query](#) property is updated with the constructed [predicate](#), and the Tree Grid is refreshed to update the changes in the UI.

On the other hand, the [actionComplete](#) event is used to manage the completion of the search operation. It ensures that the search input value is updated if necessary and clears the [query](#) when the search input is empty.

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule, FilterService, ToolbarService, } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { Column, SearchSettingsModel, ToolbarItems, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { SearchEventArgs, KeyboardEventArgs } from '@syncfusion/ej2-angular-grids';
import { Query, Predicate } from '@syncfusion/ej2-data';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService, ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data' height='270' [treeColumnIndex]='1' [searchSettings]='searchOptions'

```

```

[toolbar]='toolbarOptions' childMapping='subtasks'
(actionBegin)="actionBegin($event)"
(actionComplete)="actionComplete($event)">
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
        <e-column field='progress' headerText='Progress'
textAlign='Right' width=80></e-column>
        <e-column field='priority' headerText='Priority'
textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`
}))
export class AppComponent implements OnInit {
    public data?: object[];
    public toolbarOptions?: ToolbarItems[];
    @ViewChild('treegrid')
    public treegrid?: TreeGridComponent;
    public values?: string;
    public key = '';
    public removeQuery = false;
    public valueAssign = false;
    public searchOptions?: SearchSettingsModel;
    ngOnInit(): void {
        this.data = sampleData;
        this.toolbarOptions = ['Search'];
        this.searchOptions = {
            fields: [
                'taskID',
                'taskName',
                'duration',
                'progress',
                'priority'
            ],
            operator: 'contains',
            key: '',
            ignoreCase: true,
        };
    }
    actionBegin({ requestType, searchString }: SearchEventArgs) {
        if (requestType == 'searching') {
            const keys = (searchString as string).split(',');
            var flag = true;
            var predicate: any;
            if (keys.length > 1) {
                if ((this.treegrid as TreeGridComponent).searchSettings.key
!= '' ) {
                    this.values = searchString;
                    keys.forEach((key: string) => {
                        (this.treegrid as
TreeGridComponent).getColumns().forEach((col: Column) => {
                            if (flag) {

```

```

        predicate = new Predicate(col.field,
'contains', key, true);
        flag = false;
    }
    else {
        var predic = new Predicate(col.field,
'contains', key, true);
        predicate = predicate.or(predic);
    }
    });
    });

    (this.treegrid as TreeGridComponent).query = new
Query().where(predicate);
    (this.treegrid as TreeGridComponent).searchSettings.key =
'';

    this.valueAssign = true;
    this.removeQuery = true;
    (this.treegrid as TreeGridComponent).refresh();
    }
    }
    }
    }
    actionComplete(args: SearchEventArgs) {
        if (args.requestType === 'refresh' && this.valueAssign) {
            const searchBar = document.querySelector<HTMLInputElement>('#' +
(this.treegrid as TreeGridComponent).element.id + '_searchbar');
            if (searchBar) {
                searchBar.value = this.values || '';
                this.valueAssign = false;
            }
            else if (
                args.requestType === 'refresh' &&
                this.removeQuery
            ) {
                const searchBar = document.querySelector<HTMLInputElement>('#'
+ (this.treegrid as TreeGridComponent).element.id + '_searchbar');
                if (searchBar) {
                    searchBar.value = '';
                }
                (this.treegrid as TreeGridComponent).query = new Query();
                this.removeQuery = false;
                (this.treegrid as TreeGridComponent).refresh();
            }
        }
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,

```



```

        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
                endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
        ]
    }
]

```

```

    },
    {
      taskID: 12,
      taskName: 'Implementation Phase',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'Normal',
      approved: false,
      duration: 11,
      progress: 66,
      subtasks: [
        {
          taskID: 13,
          taskName: 'Phase 1',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/27/2017'),
          priority: 'High',
          approved: false,
          progress: 50,
          duration: 11,
          subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
              { taskID: 15, taskName: 'Development Task 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
                progress: '50', priority: 'High', approved: false },
              { taskID: 16, taskName: 'Development Task 2',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
                progress: '50', priority: 'Low', approved: true },
              { taskID: 17, taskName: 'Testing', startDate: new
                Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
                progress: '0', priority: 'Normal', approved: true },
              { taskID: 18, taskName: 'Bug fix', startDate: new
                Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
                progress: '0', priority: 'Critical', approved: false },
              { taskID: 19, taskName: 'Customer review meeting',
                startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
                progress: '0', priority: 'High', approved: false },
              { taskID: 20, taskName: 'Phase 1 complete',
                startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
                progress: '50', priority: 'Low', approved: true }
            ]
          }
        ]
      }
    ]
  },

```

```

        {
            taskID: 21,
            taskName: 'Phase 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'High',
            approved: false,
            duration: 12,
            progress: 60,
            subtasks: [{
                taskID: 22,
                taskName: 'Implementation Module 2',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/28/2017'),
                priority: 'Critical',
                approved: false,
                duration: 12,
                progress: 90,
                subtasks: [
                    { taskID: 23, taskName: 'Development Task 1',
                        startDate: new Date('02/17/2017'),
                        endDate: new Date('02/20/2017'), duration: 4,
                        progress: '50', priority: 'Normal', approved: true },
                    { taskID: 24, taskName: 'Development Task 2',
                        startDate: new Date('02/17/2017'),
                        endDate: new Date('02/20/2017'), duration: 4,
                        progress: '50', priority: 'Critical', approved: true },
                    { taskID: 25, taskName: 'Testing', startDate: new
                        Date('02/21/2017'),
                        endDate: new Date('02/24/2017'), duration: 2,
                        progress: '0', priority: 'High', approved: false },
                    { taskID: 26, taskName: 'Bug fix', startDate: new
                        Date('02/25/2017'),
                        endDate: new Date('02/26/2017'), duration: 2,
                        progress: '0', priority: 'Low', approved: false },
                    { taskID: 27, taskName: 'Customer review meeting',
                        startDate: new Date('02/27/2017'),
                        endDate: new Date('02/28/2017'), duration: 2,
                        progress: '0', priority: 'Critical', approved: true },
                    { taskID: 28, taskName: 'Phase 2 complete',
                        startDate: new Date('02/28/2017'),
                        endDate: new Date('02/28/2017'), duration: 0,
                        progress: '50', priority: 'Normal', approved: false }
                ]
            }
        ],
        {
            taskID: 29,
            taskName: 'Phase 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            approved: false,
            duration: 11,
            progress: 30,
            subtasks: [{
                taskID: 30,

```

```

        taskName: 'Implementation Module 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'High',
        approved: false,
        duration: 11,
        progress: 60,
        subtasks: [
            { taskID: 31, taskName: 'Development Task 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
            progress: '50', priority: 'Low', approved: true },
            { taskID: 32, taskName: 'Development Task 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
            progress: '50', priority: 'Normal', approved: false },
            { taskID: 33, taskName: 'Testing', startDate: new
            Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
            progress: '0', priority: 'Critical', approved: true },
            { taskID: 34, taskName: 'Bug fix', startDate: new
            Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
            progress: '0', priority: 'High', approved: false },
            { taskID: 35, taskName: 'Customer review meeting',
            startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
            progress: '0', priority: 'Normal', approved: true },
            { taskID: 36, taskName: 'Phase 3 complete',
            startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
            progress: '50', priority: 'Critical', approved: false },
        ]
    }
}
];

```

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 ignore accent while searching

By default, the searching operation in the TreeGrid component does not ignore diacritic characters or accents. However, there are cases where ignoring diacritic characters becomes necessary. This feature enhances the search experience by enabling data searching without considering accents, ensuring a more comprehensive and accurate search and it can be achieved by utilizing the [searchSettings.ignoreAccent](#) property of the TreeGrid component as **true**.

The following example demonstrates how to define the `ignoreAccent` property within the `searchSettings` property of the tree grid. Additionally, the [EJ2 Toggle Switch Button](#) component is included to modify the value of the `searchSettings.ignoreAccent` property. When the switch is toggled, the [change](#) event is triggered, and the `searchSettings.ignoreAccent` property is updated accordingly. This functionality helps to visualize the impact of the `searchSettings.ignoreAccent` setting when performing search operations.

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService, ToolbarService, } from
 '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { diacritics } from './datasource';
import { ToolbarItems, SearchSettingsModel, TreeGridComponent } from
 '@syncfusion/ej2-angular-treegrid';
import { SearchEventArgs } from '@syncfusion/ej2-angular-grids';
import { Predicate, Query } from '@syncfusion/ej2-data';
import { ChangeEventArgs } from '@syncfusion/ej2-angular-buttons';
import { SwitchModule } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [
    TreeGridAllModule, SwitchModule
  ],
  providers: [FilterService, ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<div>
    <label style="padding: 10px 10px">Enable or disable
ignoreAccent property</label>
    <ejs-switch id="switch" [checked]=true
(change)="onSwitchChange($event)"></ejs-switch>
  </div>
  <ejs-treegrid #treegrid [dataSource]='data' height='270'
[treeColumnIndex]='1' [toolbar]='toolbarOptions' childMapping='Children'>
    <e-columns>
      <e-column field='EmpID' headerText='ID'
textAlign='Right' width=90></e-column>
      <e-column field='Name' headerText='Name'
textAlign='Left' width=180></e-column>
      <e-column field='Designation'
headerText='Designation' textAlign='Right' width=150></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {

  public data?: Object[];
  public toolbarOptions?: ToolbarItems[];
  @ViewChild('treegrid') public treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = diacritics;
    this.toolbarOptions = ['Search'];
  }
  onSwitchChange(args: any) {
```

```

        if (args.checked) {
            ((this.treegrid as TreeGridComponent).grid.searchSettings as
any).ignoreAccent = true;
        } else {
            ((this.treegrid as TreeGridComponent).grid.searchSettings as
any).ignoreAccent = false;
        }
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let diacritics: Object[] = [{
    'Name': 'Aeróbics',
    'Designation': 'Chief Executive Officer',
    'EmployeeID': '1',
    'EmpID': 'EMP001',
    'Address': '507 - 20th Ave. E.Apt. 2A, Seattle',
    'Contact': '(206) 555-9857',
    'Country': 'USA',
    'DOB': new Date('2/15/1963'),

    'Children': [{
        'Name': 'Aerógrafía en Agua',
        'Designation': 'Vice President',
        'EmployeeID': '2',
        'EmpID': 'EMP004',
        'Address': '722 Moss Bay Blvd., Kirkland',
        'Country': 'USA',
        'Contact': '(206) 555-3412',
        'DOB': new Date('5/20/1971'),

        'Children': [{
            'Name': 'AerografÃa',
            'Designation': 'Marketing Executive',
            'EmployeeID': '3',
            'EmpID': 'EMP035',
            'Address': '4110 Old Redmond Rd., Redmond',
            'Country': 'USA',
            'Contact': '(206) 555-8122',
            'DOB': new Date('3/19/1966'),
            'Children': [
                {
                    'Name': 'Aeromodelaje',
                    'Designation': 'Sales Representative',
                    'EmployeeID': '4',
                    'EmpID': 'EMP045',
                    'Address': '14 Garrett Hill, London',
                    'Country': 'UK',
                    'Contact': '(71) 555-4848',
                    'DOB': new Date('9/20/1980')
                }
            ]
        }
    ]
}
],

```

```

    {
      'Name': 'Águilas',
      'Designation': 'Sales Representative',
      'EmployeeID': '5',
      'EmpID': 'EMP091',
      'Address': '4726 - 11th Ave. N.E., Seattle',
      'Country': 'USA',
      'Contact': '(206) 555-1189',
      'DOB': new Date('10/19/1989')
    },
    {
      'Name': 'Álbumes de Delta',
      'Designation': 'Sales Representative',
      'EmployeeID': '6',
      'EmpID': 'EMP110',
      'Address': 'Coventry House Miner Rd., London',
      'Country': 'UK',
      'Contact': '(71) 555-3636',
      'DOB': new Date('11/02/1987')
    },
    {
      'Name': 'ÃÄlbumes de Música',
      'Designation': 'Sales Coordinator',
      'EmployeeID': '7',
      'EmpID': 'EMP131',
      'Address': 'Edgeham Hollow Winchester Way, London',
      'Country': 'UK',
      'Contact': '(71) 555-3636',
      'DOB': new Date('11/06/1990')
    },
  ],
  {
    {
      'Name': 'Alusivos',
      'Designation': 'Sales Executive',
      'EmployeeID': '8',
      'EmpID': 'EMP039',
      'Address': '7 Houndstooth Rd., London',
      'Country': 'UK',
      'Contact': '(71) 555-3690',
      'DOB': new Date('02/02/1980'),
      'Children': [
        {
          'Name': 'ÃAerografía',
          'Designation': 'Sales Representative',
          'EmployeeID': '9',
          'EmpID': 'EMP213',
          'Address': '4726 - 11th Ave. N.E., California',
          'Country': 'USA',
          'Contact': '(206) 555-1989',
          'DOB': new Date('01/21/1986')
        },
        {
          'Name': 'Análisis de Escritura a Mano',
          'Designation': 'Sales Coordinator',
          'EmployeeID': '10',

```

```

        'EmpID': 'EMP201',
        'Address': 'Coventry House Miner Rd., London',
        'Country': 'UK',
        'Contact': '(71) 555-2222',
        'DOB': new Date('12/01/1990')
    },
    {
        'Name': 'Aeromodelaje',
        'Designation': 'Sales Representative',
        'EmployeeID': '11',
        'EmpID': 'EMP197',
        'Address': '200 Lincoln Ave, Salinas, CA 93901',
        'Country': 'USA',
        'Contact': '(1731) 758-7408',
        'DOB': new Date('03/23/1987')
    },
    {
        'Name': 'Álbumes de Delta',
        'Designation': 'Sales Representative',
        'EmployeeID': '12',
        'EmpID': 'EMP167',
        'Address': '200 Lincoln Ave, Salinas, CA 93901',
        'Country': 'USA',
        'Contact': '(1731) 758-7368',
        'DOB': new Date('08/09/1989')
    },
    ]
}
]]
]]];

```

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 set `searchSettings.ignoreAccent` property along with other search settings such as [fields](#), [operator](#), and [ignoreCase](#) to achieve the desired search behavior.

* This feature works only for the characters that are not in the ASCII range.

* This feature may have a slight impact on search performance.

Highlight the search text

The TreeGrid component allows you to visually highlight search results within the displayed data. This feature helps you to quickly identify where the search items are found within the displayed data. By adding a style to the matched text, you can quickly identify where the search items are present in the tree grid.

To achieve search text highlighting in the Tree Grid, you can utilize the [queryCellInfo](#) event. This event is triggered for each cell during the Tree Grid rendering process, allowing you to customize the cell content based on your requirements.

The following example demonstrates how to highlight search text in tree grid using the `queryCellInfo` event. The `queryCellInfo` event checks if the current cell is in the desired search column, retrieves the cell value, search keyword and uses the `includes` method to check if the cell value contains the search keyword. If it does, the matched text is replaced with the same text wrapped in a `span` tag with a `customcss` class. You can then use CSS to define the `customcss` class and style to easily identify where the search keywords are present in the tree grid.

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule, FilterService, ToolbarService, } from
 '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit ,ViewChild,ViewEncapsulation} from
 '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems, SearchSettingsModel, TreeGridComponent } from
 '@syncfusion/ej2-angular-treegrid';
import { QueryCellInfoEventArgs, SearchEventArgs } from '@syncfusion/ej2-
angular-grids';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [FilterService, ToolbarService],
  standalone: true,
  selector: 'app-container',
  encapsulation:ViewEncapsulation.None,
  template: `<ejs-treegrid #treegrid [dataSource]='data' height='270'
 [treeColumnIndex]='1' [toolbar]='toolbarOptions'
 (actionBegin)="actionBegin($event)" (queryCellInfo)="queryCellInfo($event)"
 childMapping='subtasks'>
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='priority' headerText='Priority'
textAlign='Right' width=80></e-column>
      </e-columns>
    </ejs-treegrid>`,
  styles:[`.e-rowcell .customcss{
    background-color:yellow;
  }`]
})
export class AppComponent implements OnInit {

  public key = '';
  public data?: Object[];
  public toolbarOptions?: ToolbarItems[];

  @ViewChild('treegrid') public treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
  }
}
```

```

        this.toolbarOptions = ['Search'];
    }
    actionBegin(args: SearchEventArgs) {

        if (args.requestType === 'searching') {
            (this.key as string) = (args.searchString as
string).toLowerCase();
        }
    }
    queryCellInfo(args: QueryCellInfoEventArgs) {
        if ((this.key as string) !== '') {
            var cellContent = (args.data as any)[(args.column as any).field];
            var parsedContent = cellContent.toString().toLowerCase();
            if (parsedContent.includes((this.key as string).toLowerCase())) {
                var i = 0;
                var searchStr = '';
                while (i < (this.key as string).length) {
                    var index = parsedContent.indexOf((this.key as string)[i]);
                    searchStr = searchStr + cellContent.toString()[index];
                    i++;
                }
                (args.cell as HTMLElement).innerHTML = ((args.cell as
HTMLElement).innerText as any).replaceAll(
                    searchStr,
                    "<span class='customcss'>" + searchStr + "</span>"
                );
            }
        }
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Low', approved: true },
        ]
    }
]

```

```

        { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
          endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
          endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
      ],
    },
    {
      taskID: 6,
      taskName: 'Design',
      startDate: new Date('02/10/2017'),
      endDate: new Date('02/14/2017'),
      duration: 3,
      progress: 86,
      priority: 'High',
      approved: false,
      subtasks: [
        { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
          endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
        { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
          endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
          endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
      ]
    },
  ],
  {
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
      {
        taskID: 13,
        taskName: 'Phase 1',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),

```

```

        priority: 'High',
        approved: false,
        progress: 50,
        duration: 11,
        subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
                { taskID: 15, taskName: 'Development Task 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
                progress: '50', priority: 'High', approved: false },
                { taskID: 16, taskName: 'Development Task 2',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
                progress: '50', priority: 'Low', approved: true },
                { taskID: 17, taskName: 'Testing', startDate: new
                Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
                progress: '0', priority: 'Normal', approved: true },
                { taskID: 18, taskName: 'Bug fix', startDate: new
                Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
                progress: '0', priority: 'Critical', approved: false },
                { taskID: 19, taskName: 'Customer review meeting',
                startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
                progress: '0', priority: 'High', approved: false },
                { taskID: 20, taskName: 'Phase 1 complete',
                startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
                progress: '50', priority: 'Low', approved: true }
            ]
        }]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,

```

```

        duration: 12,
        progress: 90,
        subtasks: [
            { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
            { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
            { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
            endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
            endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
            { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
            endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
        ]
    }
},
{
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
        taskID: 30,
        taskName: 'Implementation Module 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'High',
        approved: false,
        duration: 11,
        progress: 60,
        subtasks: [
            { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },

```

```

        { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
  }
}
];

```

MAIN.TS

```

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

```

Clear search by external button

The TreeGrid component provides a capability to clear searched data in the tree grid. This functionality offers the ability to reset or clear any active search filters that have been applied to the tree grid's data.

To clear the searched tree grid records from an external button, you can set the [searchSettings.key](#) property to an empty string to clear the search text. This property represents the current search text in the search box.

The following example demonstrates how to clear the searched records using an external button.

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule, FilterService, ToolbarService, } from
'@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild, ViewEncapsulation } from
'@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems, SearchSettingsModel, TreeGridComponent } from
'@syncfusion/ej2-angular-treegrid';
import { QueryCellInfoEventArgs, SearchEventArgs } from '@syncfusion/ej2-
angular-grids';
import { Predicate, Query } from '@syncfusion/ej2-data';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';

```

```

@Component({
  imports: [
    TreeGridAllModule, ButtonModule,
  ],
  standalone: true,
  providers: [FilterService, ToolbarService],
  selector: 'app-container',
  encapsulation: ViewEncapsulation.None,
  template: `<button ejs-button id='clear' (click)='clearSearch()'>Clear
Search</button>
    <ejs-treegrid #treegrid [dataSource]='data' height='230'
[treeColumnIndex]='1' [toolbar]='toolbarOptions'
[searchSettings]='searchOptions' childMapping='subtasks'>
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='priority' headerText='Priority'
textAlign='Right' width=80></e-column>
      </e-columns>
    </ejs-treegrid>`,
})
export class AppComponent implements OnInit {

  public searchOptions?: SearchSettingsModel;
  public data?: Object[];
  public toolbarOptions?: ToolbarItems[];

  @ViewChild('treegrid') public treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
    this.toolbarOptions = ['Search'];
    this.searchOptions = { fields: ['taskName'], operator: 'contains',
key: 'plan', ignoreCase: true};
  }
  clearSearch() {
    (this.treegrid as TreeGridComponent).searchSettings.key = '';
  }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date('02/03/2017'),
    endDate: new Date('02/07/2017'),
    progress: 100,
  }
]

```

```

        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
                endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
        ]
    },
    {
        taskID: 12,
        taskName: 'Implementation Phase',

```



```

    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
      {
        taskID: 13,
        taskName: 'Phase 1',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'High',
        approved: false,
        progress: 50,
        duration: 11,
        subtasks: [{
          taskID: 14,
          taskName: 'Implementation Module 1',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/27/2017'),
          priority: 'Normal',
          duration: 11,
          progress: 10,
          approved: false,
          subtasks: [
            { taskID: 15, taskName: 'Development Task 1',
              startDate: new Date('02/17/2017'),
              endDate: new Date('02/19/2017'), duration: 3,
              progress: '50', priority: 'High', approved: false },
            { taskID: 16, taskName: 'Development Task 2',
              startDate: new Date('02/17/2017'),
              endDate: new Date('02/19/2017'), duration: 3,
              progress: '50', priority: 'Low', approved: true },
            { taskID: 17, taskName: 'Testing', startDate: new
              Date('02/20/2017'),
              endDate: new Date('02/21/2017'), duration: 2,
              progress: '0', priority: 'Normal', approved: true },
            { taskID: 18, taskName: 'Bug fix', startDate: new
              Date('02/24/2017'),
              endDate: new Date('02/25/2017'), duration: 2,
              progress: '0', priority: 'Critical', approved: false },
            { taskID: 19, taskName: 'Customer review meeting',
              startDate: new Date('02/26/2017'),
              endDate: new Date('02/27/2017'), duration: 2,
              progress: '0', priority: 'High', approved: false },
            { taskID: 20, taskName: 'Phase 1 complete',
              startDate: new Date('02/27/2017'),
              endDate: new Date('02/27/2017'), duration: 0,
              progress: '50', priority: 'Low', approved: true }
          ]
        }
      ]
    },
    {
      taskID: 21,
      taskName: 'Phase 2',
      startDate: new Date('02/17/2017'),

```

```

        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
                { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
                endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
                endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
                { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
                { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
                endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
            ]
        }
    ],
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',

```

```

        approved: false,
        duration: 11,
        progress: 60,
        subtasks: [
            { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
            { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
        ]
    }
}
];

```

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 clear the searched records by using the clear icon within the search input field.

See also

- [How to perform search by using Wildcard and LIKE operator filter](#)

Paging in Angular TreeGrid component

Paging provides an option to display tree grid data in segmented pages, making it easier to navigate through large datasets. This feature is particularly useful when dealing with extensive data sets.

To enable paging, you need to set the [allowPaging](#) property to **true**. This property determines whether paging is enabled or disabled for the tree grid. When paging is enabled, a pager component rendered at the bottom of the tree grid, allowing you to navigate through different pages of data.

To use paging, you need to inject the **PageService** into the provider section of your **AppModule**. This service provides the necessary methods and events to handle paging functionality.

Paging options can be configured through the [pageSettings](#) property. The `pageSettings` object allows you to control various aspects of paging, such as the page size, current page, and total number of records.

You can achieve better performance by using tree grid paging to fetch only a pre-defined number of records from the data source.

Customize the pager options

Customizing the pager options in the Tree Grid allows you to tailor the pagination control according to your specific requirements. You can customize the pager to display the number of pages using the [pageCount](#) property, change the current page using [currentPage](#) property, display the number of records in the tree grid using the [pageSize](#) property, and even adjust the page sizes in a dropdown using the [pageSizes](#) property. Additionally, you can include the current page as a query string in the URL for convenient navigation.

Change the page size

The Tree Grid allows you to control the number of records displayed per page, providing you with flexibility in managing your data. This feature is particularly useful when you want to adjust the amount of data visible to you at any given time. To achieve this, you can utilize the [pageSettings.pageSize](#) property. This property is used to specify the initial number of records to display on each page. The default value of `pageSize` property is **12**.

The following example demonstrates how to change the page size of a tree grid using an external button click based on **TextBox** input.

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { PageService, TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { TextBoxModule } from '@syncfusion/ej2-angular-inputs';
import { Component, OnInit, ViewChild, ViewEncapsulation } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, PageSettingsModel } from '@syncfusion/ej2-angular-treegrid';
import { TextBoxComponent } from '@syncfusion/ej2-angular-inputs';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [
    TreeGridAllModule, ButtonModule, TextBoxModule
  ],
  providers: [PageService],
  standalone: true,
  selector: 'app-container',
  encapsulation: ViewEncapsulation.None,
```

```

    template: `<div>
        <label style="padding: 30px 17px 0 0">Enter page
size:</label>
        <ejs-textbox #textbox width="120"></ejs-textbox>
        <button ej-button #button id="button"
(created)=clickHandler($event)>click button</button>
        </div>
        <ejs-treegrid #treegrid [dataSource]='data' height=250
[treeColumnIndex]='1' [allowPaging]='true' [pageSettings]='pageSettings'
childMapping='subtasks' >
            <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
                <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
                <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
            </e-columns>
        </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public pageSettings?: Object ;
        @ViewChild('textbox') public textbox?: TextBoxComponent;
        @ViewChild('button') public button?: ButtonComponent;
        @ViewChild('treegrid')
        public treegrid?: TreeGridComponent;
        ngOnInit(): void {
            this.data = sampleData;
        }
        clickHandler(args:any): void {
            (this.button as ButtonComponent).element.addEventListener('click', (e:
MouseEvent) => {
                e.preventDefault(); // Prevent any default behavior of the button
click
                (this.treegrid as TreeGridComponent).pageSettings.pageSize =
parseInt((this.textbox as TextBoxComponent).value, 10);
            });
        }
    }

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
    }
]

```

```

        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ],
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
                endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
        ],
    },
    {
        taskID: 12,
        taskName: 'Implementation Phase',

```

```

    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
      {
        taskID: 13,
        taskName: 'Phase 1',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'High',
        approved: false,
        progress: 50,
        duration: 11,
        subtasks: [{
          taskID: 14,
          taskName: 'Implementation Module 1',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/27/2017'),
          priority: 'Normal',
          duration: 11,
          progress: 10,
          approved: false,
          subtasks: [
            { taskID: 15, taskName: 'Development Task 1',
              startDate: new Date('02/17/2017'),
              endDate: new Date('02/19/2017'), duration: 3,
              progress: '50', priority: 'High', approved: false },
            { taskID: 16, taskName: 'Development Task 2',
              startDate: new Date('02/17/2017'),
              endDate: new Date('02/19/2017'), duration: 3,
              progress: '50', priority: 'Low', approved: true },
            { taskID: 17, taskName: 'Testing', startDate: new
              Date('02/20/2017'),
              endDate: new Date('02/21/2017'), duration: 2,
              progress: '0', priority: 'Normal', approved: true },
            { taskID: 18, taskName: 'Bug fix', startDate: new
              Date('02/24/2017'),
              endDate: new Date('02/25/2017'), duration: 2,
              progress: '0', priority: 'Critical', approved: false },
            { taskID: 19, taskName: 'Customer review meeting',
              startDate: new Date('02/26/2017'),
              endDate: new Date('02/27/2017'), duration: 2,
              progress: '0', priority: 'High', approved: false },
            { taskID: 20, taskName: 'Phase 1 complete',
              startDate: new Date('02/27/2017'),
              endDate: new Date('02/27/2017'), duration: 0,
              progress: '50', priority: 'Low', approved: true }
          ]
        }
      ]
    },
    {
      taskID: 21,
      taskName: 'Phase 2',
      startDate: new Date('02/17/2017'),

```

```

        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
                { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
                endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
                endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
                { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
                { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
                endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
            ]
        }
    ],
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',

```



```

        approved: false,
        duration: 11,
        progress: 60,
        subtasks: [
            { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
            { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
        ]
    }
}
];

```

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 page count

The Tree Grid allows you to adjust the number of pages displayed in the pager container. This is useful when you want to manage the number of pages you see while navigating through extensive datasets. The default value of [pageCount](#) property is 8.

To change the page count in the Tree Grid, you can utilize the `pageCount` property of [pageSettings](#), which defines the number of pages displayed in the pager container.

The following example demonstrates how to change the page count of a tree grid using an external button click based on **TextBox** input.

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { PageService, TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { TextBoxModule } from '@syncfusion/ej2-angular-inputs';
import { Component, OnInit, ViewChild, ViewEncapsulation } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, PageSettingsModel } from '@syncfusion/ej2-angular-treegrid';
import { TextBoxComponent } from '@syncfusion/ej2-angular-inputs';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [
    TreeGridAllModule, ButtonModule, TextBoxModule
  ],
  providers: [PageService],
  standalone: true,
  selector: 'app-container',
  encapsulation: ViewEncapsulation.None,
  template: `<div>
    <label style="padding: 30px 17px 0 0">Enter page
count:</label>
    <ejs-textbox #textbox width="120"></ejs-textbox>
    <button ejs-button #button id="button"
(created)=clickHandler($event)>click button</button>
    </div>
    <ejs-treegrid #treegrid [dataSource]='data' height=250
[treeColumnIndex]='1' [allowPaging]='true' [pageSettings]='pageSettings'
childMapping='subtasks' >
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
      </e-columns>
    </ejs-treegrid>`
  })
export class AppComponent implements OnInit {
  public data?: Object[];
  public pageSettings?: Object ;
  @ViewChild('textbox') public textbox?: TextBoxComponent;
  @ViewChild('button') public button?: ButtonComponent;
  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
    this.pageSettings={pageSize:8};
  }
  clickHandler(args:any): void {

```

```

        (this.button as ButtonComponent).element.addEventListener('click', (e:
MouseEvent) => {
            e.preventDefault(); // Prevent any default behavior of the button
click
            (this.treegrid as TreeGridComponent).pageSettings.pageCount =
parseInt((this.textbox as TextBoxComponent).value, 10);
        });
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ],
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),

```

```

        endDate: new Date('02/12/2017'), duration: 3, progress: 60,
        priority: 'Normal', approved: false },
        { taskID: 8, taskName: 'Develop prototype', startDate: new
        Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
        priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
        new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
        Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
        Date('02/14/2017'),
        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
        priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            progress: 50,
            duration: 11,
            subtasks: [{
                taskID: 14,
                taskName: 'Implementation Module 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'Normal',
                duration: 11,
                progress: 10,
                approved: false,
                subtasks: [
                    { taskID: 15, taskName: 'Development Task 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'High', approved: false },
                    { taskID: 16, taskName: 'Development Task 2',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'Low', approved: true },
                ]
            }
        ]
    ]
}

```

```

        { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
      taskID: 22,
      taskName: 'Implementation Module 2',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/28/2017'),
      priority: 'Critical',
      approved: false,
      duration: 12,
      progress: 90,
      subtasks: [
        { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
        { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),

```

```

        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
                { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
                { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
            ]
        }
    ]
}
    ]
}

```

```
];
```

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 current page

The Tree Grid allows you to change the currently displayed page, which can be particularly useful when you need to navigate through different pages of data either upon the initial rendering of the tree grid or update the displayed page based on interactions or specific conditions. The default value of [currentPage](#) property is 1.

To change the current page in the Tree Grid, you can utilize the `currentPage` property of [pageSettings](#), which defines the current page number of the pager.

The following example demonstrates how to dynamically change the current page using an external button click based on **TextBox** input:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { TextBoxModule } from '@syncfusion/ej2-angular-inputs';
import { Component, OnInit, ViewChild, ViewEncapsulation } from
 '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, PageSettingsModel } from '@syncfusion/ej2-
angular-treegrid';
import { TextBoxComponent } from '@syncfusion/ej2-angular-inputs';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [
    TreeGridAllModule, ButtonModule, TextBoxModule
  ],
  providers:[PageService],
  standalone: true,
  selector: 'app-container',
  encapsulation:ViewEncapsulation.None,
  template: `<div>
    <label style="padding: 30px 17px 0 0">Enter current
page:</label>
    <ejs-textbox #textbox width="120"></ejs-textbox>
    <button ejs-button #button id="button"
(created)=clickHandler($event)>click button</button>
    </div>
    <ejs-treegrid #treegrid [dataSource]='data' height=250
[treeColumnIndex]='1' [allowPaging]='true' [pageSettings]='pageSettings'
childMapping='subtasks' >
    <e-columns>
```

```

        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public pageSettings?: Object ;
        @ViewChild('textbox') public textbox?: TextBoxComponent;
        @ViewChild('button') public button?: ButtonComponent;
        @ViewChild('treegrid')
        public treegrid?: TreeGridComponent;
        ngOnInit(): void {
            this.data = sampleData;
            this.pageSettings={pageSize:8};
        }
        clickHandler(args:any): void {
            (this.button as ButtonComponent).element.addEventListener('click', (e:
MouseEvent) => {
                e.preventDefault(); // Prevent any default behavior of the button
click
                (this.treegrid as TreeGridComponent).pageSettings.currentPage =
parseInt((this.textbox as TextBoxComponent).value, 10);
            });
        }
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),

```



```

        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
        { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ]
},
{
    taskID: 6,
    taskName: 'Design',
    startDate: new Date('02/10/2017'),
    endDate: new Date('02/14/2017'),
    duration: 3,
    progress: 86,
    priority: 'High',
    approved: false,
    subtasks: [
        { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
        { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',

```

```

        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'High',
        approved: false,
        progress: 50,
        duration: 11,
        subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
                { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
                { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
                { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
            ]
        }
    ]
},
{
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
        taskID: 22,
        taskName: 'Implementation Module 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),

```

```

        priority: 'Critical',
        approved: false,
        duration: 12,
        progress: 90,
        subtasks: [
            { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
            { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
            { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
            endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
            endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
            { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
            endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
        ]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),

```

```

        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
  }
}
]
};

```

MAIN.TS

```

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

```

Add current page in URL as a query string

The Tree Grid allows you to include the current page information as a query string in the URL. This feature is particularly useful for scenarios where you need to maintain and share the state of the tree grid's pagination.

To add the current page detail to the URL as a query string in the tree grid, you can enable the [enableQueryString](#) property. When this property is set to **true**, it will automatically pass the current page information as a query string parameter along with the URL when navigating to other pages within the tree grid.

By enabling the `enableQueryString` property, you can easily copy the URL of the current page and share it with others. When the shared URL is opened, it will load the tree grid with the exact page that was originally shared.

In the following example, the [EJ2 Toggle Switch Button](#) component is added to enable or disable the addition of the current page to the URL as a query string. When the switch is toggled, the [change](#) event is triggered and the `enableQueryString` property of the tree grid is updated accordingly.

APP.COMPONENT.TS

```

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

```

```

import { TreeGridAllModule, PageService } from '@syncfusion/ej2-angular-treegrid';
import { SwitchModule } from '@syncfusion/ej2-angular-buttons';
import { Component, OnInit, ViewChild, ViewEncapsulation } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, PageSettingsModel } from '@syncfusion/ej2-angular-treegrid';
import { SwitchComponent } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [
    TreeGridAllModule, SwitchModule
  ],
  providers:[PageService],
  standalone: true,
  selector: 'app-container',
  encapsulation:ViewEncapsulation.None,
  template: `<div style="padding: 20px 0px 20px 0px">
    <label>Enable/Disable Query String</label>
    <ejs-switch #switch id="switch"
[ (checked) ]="enableQuery" (change)="toggleQueryString($event)"></ejs-switch>
    </div>
    <ejs-treegrid #treegrid [dataSource]='data' height=230
[treeColumnIndex]='1' [allowPaging]='true' [pageSettings]='pageSettings'
childMapping='subtasks' >
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
      </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pageSettings?: Object ;
  public initialPage?: object;
  @ViewChild('switch') public switch?: SwitchComponent;
  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  public enableQuery = true;
  ngOnInit(): void {
    this.data = sampleData;
    this.pageSettings={pageSize:8,enableQueryString:this.enableQuery};
  }
  toggleQueryString(args:any): void {
    (this.treegrid as TreeGridComponent).pageSettings.enableQueryString =
args.checked;
  }
}

```

DATASOURCE.TS

```
/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date('02/03/2017'),
    endDate: new Date('02/07/2017'),
    progress: 100,
    duration: 5,
    priority: 'Normal',
    approved: false,
    subtasks: [
      { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
        priority: 'Normal', approved: false },
      { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
        priority: 'Low', approved: true },
      { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
        priority: 'Critical', approved: false },
      { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
        priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 6,
    taskName: 'Design',
    startDate: new Date('02/10/2017'),
    endDate: new Date('02/14/2017'),
    duration: 3,
    progress: 86,
    priority: 'High',
    approved: false,
    subtasks: [
      { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 60,
        priority: 'Normal', approved: false },
      { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
        priority: 'Critical', approved: false },
      { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'Low', approved: true },
    ]
  }
]
```

```

        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
          endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
      ]
    },
    {
      taskID: 12,
      taskName: 'Implementation Phase',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'Normal',
      approved: false,
      duration: 11,
      progress: 66,
      subtasks: [
        {
          taskID: 13,
          taskName: 'Phase 1',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/27/2017'),
          priority: 'High',
          approved: false,
          progress: 50,
          duration: 11,
          subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
              { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
              { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
              { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
              { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
              { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),

```

```

        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
  }
},
{
  taskID: 21,
  taskName: 'Phase 2',
  startDate: new Date('02/17/2017'),
  endDate: new Date('02/28/2017'),
  priority: 'High',
  approved: false,
  duration: 12,
  progress: 60,
  subtasks: [{
    taskID: 22,
    taskName: 'Implementation Module 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'Critical',
    approved: false,
    duration: 12,
    progress: 90,
    subtasks: [
      { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
      { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
      { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
      { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
      { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
      { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  }
}
},
{
  taskID: 29,

```



```

        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'Normal', approved: false },
                { taskID: 33, taskName: 'Testing', startDate: new
                    Date('02/20/2017'),
                    endDate: new Date('02/21/2017'), duration: 2,
                    progress: '0', priority: 'Critical', approved: true },
                { taskID: 34, taskName: 'Bug fix', startDate: new
                    Date('02/24/2017'),
                    endDate: new Date('02/25/2017'), duration: 2,
                    progress: '0', priority: 'High', approved: false },
                { taskID: 35, taskName: 'Customer review meeting',
                    startDate: new Date('02/26/2017'),
                    endDate: new Date('02/27/2017'), duration: 2,
                    progress: '0', priority: 'Normal', approved: true },
                { taskID: 36, taskName: 'Phase 3 complete',
                    startDate: new Date('02/27/2017'),
                    endDate: new Date('02/27/2017'), duration: 0,
                    progress: '50', priority: 'Critical', approved: false },
            ]
        }]
    }
]
];

```

MAIN.TS

```

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

```

Page size Mode

In the TreeGrid component, the page size mode feature allows you to specify the number of records displayed on each page using the [pageSizeMode](#) property of the [pageSettings](#).

Two page size modes are available in the Tree Grid paging, each providing a different way to determine the number of records displayed on a page. Following are the two types of [pageSizeMode](#):

- **All** : This is the default mode. In this mode, the number of records displayed on each page is based on the [pageSize](#) property. All records, including both parent and child records, are considered when determining the number of records per page.
- **Root** : In this mode, the number of root nodes or the 0th level records displayed per page is based on the [pageSize](#) property. Only the root-level records are counted irrespective of the child records count, when determining the number of records per page.

By configuring the [pageSizeMode](#) property, you can control how records are paginated in the Tree Grid, providing flexibility in displaying data according to your requirements.

The following example demonstrates how to use the [pageSizeMode](#) property in the tree grid:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from '../datasource';
import { TreeGridComponent, PageService } from '@syncfusion/ej2-angular-treegrid';
import { ChangeEventArgs } from '@syncfusion/ej2-angular-dropdowns';
@Component({
  imports: [
    TreeGridAllModule, DropDownListAllModule
  ],
  providers: [PageService],
  standalone: true,
  selector: 'app-container',
  template: `<div style="display: flex">
    <label style="padding: 30px 17px 0 0;">Select page size
mode :</label>
    <ejs-dropdownlist style="padding: 26px 0 0 0" index="0"
width="100" [dataSource]="Datamode" (change)
="changepagesizemode($event)"></ejs-dropdownlist>
    <div>
      <ejs-treegrid #treegrid [dataSource]='data' height='210'
[treeColumnIndex]='1' [allowPaging]='true' [pageSettings]='pageSettings'
childMapping='subtasks' >
        <e-columns>
          <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
          <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
          <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        </e-columns>
      </ejs-treegrid>
    </div>
  </div>`
})
export class AppComponent {
  @ViewChild(TreeGridComponent) treegrid: TreeGridComponent;
  datamode: any;
  subtasks: any;
  ngOnInit(): void {
    this.datamode = sampleData;
    this.subtasks = sampleData.subtasks;
  }
}
```

```

        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public pageSettings?: Object ;
        public Datamode: Object[] = [
            { text: 'Root', value: 'Root' },
            { text: 'All', value: 'All' },

        ];
        @ViewChild('treegrid')
        public treegrid?:TreeGridComponent;
        public changepagesizemode(args: ChangeEventArgs): void {
            (this.treegrid as TreeGridComponent).pageSettings.pageSizeMode=
args.value as any;

        }
        ngOnInit(): void {
            this.data = sampleData;
            this.pageSettings = {pageSize: 2, pageSizeMode: 'Root'};
        }
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),

```

```

        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ],
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
                endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
        ]
    },
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,

```

```

        taskName: 'Implementation Module 1',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        duration: 11,
        progress: 10,
        approved: false,
        subtasks: [
            { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
            { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
            { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),

```

```

        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
        { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  },
  {
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
      taskID: 30,
      taskName: 'Implementation Module 3',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'High',
      approved: false,
      duration: 11,
      progress: 60,
      subtasks: [
        { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
        { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
          endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },

```

```

        { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
  }
}
];

```

MAIN.TS

```

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

```

Pager template

The pager template in the TreeGrid component allows you to customize the appearance and behavior of the pager element, which is used for navigation through different pages of tree grid data. This feature is particularly useful when you want to use custom elements inside the pager instead of the default elements.

To use the pager template, you need to specify the [pagerTemplate](#) property in your Tree Grid configuration. The [pagerTemplate](#) property allows you to define a custom template for the pager. Within the template, you can access the [currentPage](#), [pageSize](#), [pageCount](#), [totalPage](#) and [totalRecordCount](#) values.

The following example demonstrates how to render a **NumericTextBox** component in the pager using the [pagerTemplate](#) property:

APP.COMPONENT.TS

```

{% raw %}
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit,ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { PageSettingsModel, TreeGridComponent, PageService } from
'@syncfusion/ej2-angular-treegrid';
import { ChangeEventArgs } from '@syncfusion/ej2-inputs';
import { NumericTextBoxModule } from '@syncfusion/ej2-angular-inputs';
@Component({
  imports: [
    TreeGridAllModule, NumericTextBoxModule

```

```

],
providers:[PageService],
standalone: true,
selector: 'app-container',
template: `<ejs-treegrid #treegrid [dataSource]='data' height=250
[treeColumnIndex]='1' [allowPaging]='true' [pageSettings]='PageSettings'
childMapping='subtasks' >
<ng-template #pagerTemplate let-data>
<div class="e-pagertemplate">
<div class="col-lg-12 control-section">
<div class="content-wrapper" style="margin-top:5px;margin-left:30px;border:
none; display: inline-block ">
<ejs-numerictextbox step='1' width='75' min='1' max='3'
value={{data.currentPage}} (change)='change($event)'></ejs-numerictextbox>
</div>
</div>
<div id="totalPages" class="e-pagertemplatemessage"
style="margin-top:5px;margin-left:30px;border: none; display: inline-block
">
<span class="e-pagenomsg"> {{data.currentPage}} of {{data.totalPages}} pages
({{data.totalRecordsCount}} items)</span>
</div>
</div>
</ng-template>
<e-columns>
<e-column field='taskID' headerText='Task ID' textAlign='Right'
width=90></e-column>
<e-column field='taskName' headerText='Task Name' textAlign='Left'
width=180></e-column>
<e-column field='startDate' headerText='Start Date' textAlign='Right'
format='yMd' width=90></e-column>
<e-column field='duration' headerText='Duration' textAlign='Right'
width=80></e-column>
</e-columns>
</ejs-treegrid>`
)})
export class AppComponent implements OnInit {
public data?: Object[];
@ViewChild('treegrid')
public treeGridObj?: TreeGridComponent;
public PageSettings?: PageSettingsModel;
ngOnInit(): void {
this.data = sampleData;
}
change(args:ChangeEventArgs){
this.PageSettings = { currentPage: args.value };
}
}
{% endraw %}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [

```



```

{
  taskID: 1,
  taskName: 'Planning',
  startDate: new Date('02/03/2017'),
  endDate: new Date('02/07/2017'),
  progress: 100,
  duration: 5,
  priority: 'Normal',
  approved: false,
  subtasks: [
    { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
      endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
    { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
      endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
    { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
      endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
    { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
      endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
  ],
},
{
  taskID: 6,
  taskName: 'Design',
  startDate: new Date('02/10/2017'),
  endDate: new Date('02/14/2017'),
  duration: 3,
  progress: 86,
  priority: 'High',
  approved: false,
  subtasks: [
    { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
      endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
    { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
      endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
    { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
      endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
    { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
      endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
    { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),

```

```

        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
    ],
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,
                    taskName: 'Implementation Module 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'Normal',
                    duration: 11,
                    progress: 10,
                    approved: false,
                    subtasks: [
                        { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
                        { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                        { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
                        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
                            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
                    ]
                }
            ]
        }
    ]
}

```

```

    ]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/20/2017'), duration: 4,
                    progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/20/2017'), duration: 4,
                    progress: '50', priority: 'Critical', approved: true },
                { taskID: 25, taskName: 'Testing', startDate: new
                    Date('02/21/2017'),
                    endDate: new Date('02/24/2017'), duration: 2,
                    progress: '0', priority: 'High', approved: false },
                { taskID: 26, taskName: 'Bug fix', startDate: new
                    Date('02/25/2017'),
                    endDate: new Date('02/26/2017'), duration: 2,
                    progress: '0', priority: 'Low', approved: false },
                { taskID: 27, taskName: 'Customer review meeting',
                    startDate: new Date('02/27/2017'),
                    endDate: new Date('02/28/2017'), duration: 2,
                    progress: '0', priority: 'Critical', approved: true },
                { taskID: 28, taskName: 'Phase 2 complete',
                    startDate: new Date('02/28/2017'),
                    endDate: new Date('02/28/2017'), duration: 0,
                    progress: '50', priority: 'Normal', approved: false }
            ]
        }]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
    }
]

```

```

        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'Normal', approved: false },
                { taskID: 33, taskName: 'Testing', startDate: new
                    Date('02/20/2017'),
                    endDate: new Date('02/21/2017'), duration: 2,
                    progress: '0', priority: 'Critical', approved: true },
                { taskID: 34, taskName: 'Bug fix', startDate: new
                    Date('02/24/2017'),
                    endDate: new Date('02/25/2017'), duration: 2,
                    progress: '0', priority: 'High', approved: false },
                { taskID: 35, taskName: 'Customer review meeting',
                    startDate: new Date('02/26/2017'),
                    endDate: new Date('02/27/2017'), duration: 2,
                    progress: '0', priority: 'Normal', approved: true },
                { taskID: 36, taskName: 'Phase 3 complete',
                    startDate: new Date('02/27/2017'),
                    endDate: new Date('02/27/2017'), duration: 0,
                    progress: '50', priority: 'Critical', approved: false },
            ]
        }]
    }
]
};

```

MAIN.TS

```

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

```

Pager with page size dropdown

The pager with a page size dropdown in the Tree Grid allows you to dynamically change the number of records displayed in the tree grid. This feature is useful when you want to easily customize the number of records to be shown per page.

To enable the page size Dropdown feature in the tree grid, you need to set the [pageSettings.pageSizes](#) property to **true** in the tree grid configuration. This property configuration triggers the rendering of a dropdown list within the pager, allowing you to select the desired page size. The selected page size determines the number of records displayed on each page of the tree grid.

The following example that demonstrates how to integrate the page size Dropdown feature by configuring the `pageSizes` property:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild, ViewEncapsulation } from
 '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, PageSettingsModel, PageService } from
 '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [PageService],
  standalone: true,
  selector: 'app-container',
  encapsulation: ViewEncapsulation.None,
  template: `<ejs-treegrid #treegrid [dataSource]='data' height=250
[treeColumnIndex]='1' [allowPaging]='true' [pageSettings]='pageSettings'
childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pageSettings?: Object ;

  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
    this.pageSettings = { pageSizes: true, pageSize: 12 };
  }
}
```

DATASOURCE.TS

```
/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date('02/03/2017'),
    endDate: new Date('02/07/2017'),
    progress: 100,
    duration: 5,
    priority: 'Normal',
    approved: false,
    subtasks: [
      { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
      { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
      { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
      { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 6,
    taskName: 'Design',
    startDate: new Date('02/10/2017'),
    endDate: new Date('02/14/2017'),
    duration: 3,
    progress: 86,
    priority: 'High',
    approved: false,
    subtasks: [
      { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
      { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
      { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
      { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
```

```

        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            progress: 50,
            duration: 11,
            subtasks: [{
                taskID: 14,
                taskName: 'Implementation Module 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'Normal',
                duration: 11,
                progress: 10,
                approved: false,
                subtasks: [
                    { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
                    { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                    { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                    endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                    { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                    endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
                    { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                    endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },

```

```

        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
                { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
                endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
                endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
                { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
                { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
                endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
            ]
        }
    ]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),

```



```

        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
                { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
                { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
            ]
        }
    ]
}
];

```

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 `pageSizes` property is set to a boolean value like 'true' the page size dropdown defaults to an array of strings containing options such as ['All', '5', '10', '15', '20'].

Customize page size dropdown

The Tree Grid allows you to customize the default values of the page size dropdown in the pager, allowing you to change the number of records displayed per page. To achieve this, you can define the [pageSizes](#) property as an array of string instead of boolean value.

The following example demonstrate how to customize the default values of the pager dropdown using the `pageSizes` property:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild, ViewEncapsulation } from
 '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, PageSettingsModel, PageService } from
 '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  standalone: true,
  providers:[PageService],
  selector: 'app-container',
  encapsulation:ViewEncapsulation.None,
  template: `<ejs-treegrid #treegrid [dataSource]='data' height=250
[treeColumnIndex]='1' [allowPaging]='true' [pageSettings]='pageSettings'
childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pageSettings?: Object ;

  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
    this.pageSettings = { pageSizes: ['5', '10', '15', '20', 'All'], };
  }
}
```

DATASOURCE.TS

```
/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date('02/03/2017'),
    endDate: new Date('02/07/2017'),
    progress: 100,
    duration: 5,
    priority: 'Normal',
    approved: false,
    subtasks: [
      { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
      { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
      { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
      { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 6,
    taskName: 'Design',
    startDate: new Date('02/10/2017'),
    endDate: new Date('02/14/2017'),
    duration: 3,
    progress: 86,
    priority: 'High',
    approved: false,
    subtasks: [
      { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
      { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
      { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
      { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
```

```

        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            progress: 50,
            duration: 11,
            subtasks: [{
                taskID: 14,
                taskName: 'Implementation Module 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'Normal',
                duration: 11,
                progress: 10,
                approved: false,
                subtasks: [
                    { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
                    { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                    { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                    endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                    { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                    endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
                    { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                    endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },

```

```

        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
                { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
                endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
                endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
                { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
                { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
                endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
            ]
        }
    ]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),

```

```

        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
                { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
                { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
            ]
        }
    ]
}
];

```

MAIN.TS

```

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

```

The `pageSizes` property can be configured with either an array of strings or a boolean value.

How to navigate to particular page

Navigating to a particular page in the Tree Grid is particularly useful when dealing with large datasets. It provides a quick and efficient way to jump to a specific page within the tree grid.

To achieve page navigation, you can use the [goToPage](#) method provided by the tree grid. This method allows you to programmatically navigate to a specific page within the tree grid.

The following example demonstrates how to dynamically navigate to a particular page using the [goToPage](#) method triggered by an external button click based on **TextBox** input:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { TextBoxModule } from '@syncfusion/ej2-angular-inputs';
import { Component, OnInit, ViewChild, ViewEncapsulation } from
 '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, PageSettingsModel, PageService } from
 '@syncfusion/ej2-angular-treegrid';
import { TextBoxComponent } from '@syncfusion/ej2-angular-inputs';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [
    TreeGridAllModule, ButtonModule, TextBoxModule
  ],
  providers:[PageService],
  standalone: true,
  selector: 'app-container',
  encapsulation:ViewEncapsulation.None,
  template: `<div>
    <label style="padding: 30px 17px 0 0">Enter page
index:</label>
    <ejs-textbox #textbox width="120"></ejs-textbox>
    <button ejs-button #button id="button"
(created)=clickHandler($event)>click button</button>
    </div>
    <ejs-treegrid #treegrid [dataSource]='data' height=250
[treeColumnIndex]='1' [allowPaging]='true' [pageSettings]='pageSettings'
childMapping='subtasks' >
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
      </e-columns>
    </ejs-treegrid>`
  })
export class AppComponent implements OnInit {
  public data?: Object[];
  public pageSettings?: Object ;
```

```

    @ViewChild('treegrid')
    public treegrid?: TreeGridComponent;
    @ViewChild('textbox') public textbox?: TextBoxComponent;
    @ViewChild('button') public button?: ButtonComponent;
    ngOnInit(): void {
        this.data = sampleData;
    }
    clickHandler(args:any): void {
        (this.button as ButtonComponent).element.addEventListener('click',
(e: MouseEvent) => {
            e.preventDefault(); // Prevent any default behavior of the
button click
            (this.treegrid as
TreeGridComponent).pagerModule.goToPage(parseInt((this.textbox as
TextBoxComponent).value, 10));
        });
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ]
    },
    {

```



```

        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
                priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
                priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
                priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
                priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
                endDate: new Date('02/14/2017'), duration: 0, progress: 0,
                priority: 'Normal', approved: true }
        ],
        {
            taskID: 12,
            taskName: 'Implementation Phase',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            approved: false,
            duration: 11,
            progress: 66,
            subtasks: [
                {
                    taskID: 13,
                    taskName: 'Phase 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'High',
                    approved: false,
                    progress: 50,
                    duration: 11,
                    subtasks: [{
                        taskID: 14,
                        taskName: 'Implementation Module 1',
                        startDate: new Date('02/17/2017'),
                        endDate: new Date('02/27/2017'),
                        priority: 'Normal',
                        duration: 11,

```

```

        progress: 10,
        approved: false,
        subtasks: [
            { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
            { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
            { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
            ]
        }
    ]
}

```

```

        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  },
  {
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
      taskID: 30,
      taskName: 'Implementation Module 3',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'High',
      approved: false,
      duration: 11,
      progress: 60,
      subtasks: [
        { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
        { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),

```

```

                                endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                                { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
                                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
                                ]
                                }
                                ]
                                }
                                ]
];

```

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 get the pager element

You can get pager element in the Tree Grid. This allows you to customize the pager's appearance or behavior to meet the requirements of your application.

[getPager](#)- This method allows you to obtain a reference to the pager element within the tree grid. It returns an HTML element representing the pager.

```
`ts
```

```
this.treegrid.getPager()
```

```
,
```

Dynamically calculate page size based on element height

You have an option to dynamically calculate the page size of a tree grid by considering the height of its parent element. This functionality proves invaluable in ensuring that the tree grid's content remains within the available space, preventing the need for excessive scrolling. It primarily serves the purpose of automatically adjusting the [pageSize](#) when the height of the tree grid's parent element changes dynamically. Upon each alteration in the parent element's height, invoking this method will compute the tree grid's [pageSize](#) and present the current page records accordingly. This feature effectively addresses situations where a static [pageSize](#) value does not cater to the varying heights of different parent elements, preventing any unwanted empty spaces within the tree grid.

To achieve page size calculation based on an element's height in the tree grid, you can utilize the [calculatePageSizeByParentHeight](#) method of the grid object through the tree grid instance. This method calculates the page size based on the height of the parent element.

The following example demonstrates how to calculate the page size based on element height using the [calculatePageSizeByParentHeight](#) method triggered by a change event based on the **NumericTextBox** input:

APP.COMPONENT.TS

```

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

```

```

import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { NumericTextBoxModule } from '@syncfusion/ej2-angular-inputs';
import { Component, OnInit, ViewChild, ViewEncapsulation } from
 '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, PageSettingsModel, PageService } from
 '@syncfusion/ej2-angular-treegrid';
import { ChangeEventArgs, NumericTextBoxComponent } from '@syncfusion/ej2-
angular-inputs';
@Component({
  imports: [
    TreeGridAllModule, NumericTextBoxModule
  ],
  providers: [PageService],
  standalone: true,
  selector: 'app-container',
  encapsulation: ViewEncapsulation.None,
  template: `<div style="padding: 0 0 20px 0">
    <label style="padding: 30px 17px 0 0">Select page
size:</label>
    <ejs-numerictextbox #numericTextbox placeholder='select
container height' format='###.##' min=150 step="50"
(change)='calculatePageSize($event)' width="200px"></ejs-numerictextbox>
    </div>
    <ejs-treegrid #treegrid [dataSource]='data' height=230
[treeColumnIndex]='1' [allowPaging]='true' [pageSettings]='pageSettings'
childMapping='subtasks' >
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
      </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pageSettings?: Object;

  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  @ViewChild('numericTextbox') public numericTextbox?:
NumericTextBoxComponent;
  ngOnInit(): void {
    this.data = sampleData;
  }
  calculatePageSize({ value }: ChangeEventArgs) {
    (this.treegrid as TreeGridComponent).pageSettings.pageSize =
(this.treegrid as
TreeGridComponent).grid.calculatePageSizeByParentHeight((value as
number).toString());
  }
}

```

```
}
}
```

DATASOURCE.TS

```
/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date('02/03/2017'),
    endDate: new Date('02/07/2017'),
    progress: 100,
    duration: 5,
    priority: 'Normal',
    approved: false,
    subtasks: [
      { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
      { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
      { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
      { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 6,
    taskName: 'Design',
    startDate: new Date('02/10/2017'),
    endDate: new Date('02/14/2017'),
    duration: 3,
    progress: 86,
    priority: 'High',
    approved: false,
    subtasks: [
      { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
      { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
    ]
  }
]
```

```

        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
          endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
      ]
    },
    {
      taskID: 12,
      taskName: 'Implementation Phase',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'Normal',
      approved: false,
      duration: 11,
      progress: 66,
      subtasks: [
        {
          taskID: 13,
          taskName: 'Phase 1',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/27/2017'),
          priority: 'High',
          approved: false,
          progress: 50,
          duration: 11,
          subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
              { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
              { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
              { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
              { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),

```

```

        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
      taskID: 22,
      taskName: 'Implementation Module 2',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/28/2017'),
      priority: 'Critical',
      approved: false,
      duration: 12,
      progress: 90,
      subtasks: [
        { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
        { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
          endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
          endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
          endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
          endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
      ]
    }
  ]
}

```



```

    ]]
  },
  {
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
      taskID: 30,
      taskName: 'Implementation Module 3',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'High',
      approved: false,
      duration: 11,
      progress: 60,
      subtasks: [
        { taskID: 31, taskName: 'Development Task 1',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
          progress: '50', priority: 'Low', approved: true },
        { taskID: 32, taskName: 'Development Task 2',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
          progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
          Date('02/20/2017'),
          endDate: new Date('02/21/2017'), duration: 2,
          progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
          Date('02/24/2017'),
          endDate: new Date('02/25/2017'), duration: 2,
          progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
          startDate: new Date('02/26/2017'),
          endDate: new Date('02/27/2017'), duration: 2,
          progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
          startDate: new Date('02/27/2017'),
          endDate: new Date('02/27/2017'), duration: 0,
          progress: '50', priority: 'Critical', approved: false },
      ]
    }
  ]
}
];

```

MAIN.TS

```

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

```

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

Render pager at the top of the tree grid

The TreeGrid component provides built-in support for rendering a pager at the bottom of the tree grid by default. However, in certain scenarios, you might want to display the pager at the top of the tree grid. This can be achieved by utilizing the [dataBound](#) event. This event is triggered when the tree grid completes rendering its data. By handling this event, you can customize the rendering of the pager and move it to the top of the tree grid.

Here's an example that demonstrates how to render the pager at the top of the tree grid using the [dataBound](#) event:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild, ViewEncapsulation } from
 '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, ToolbarItems, PageSettingsModel, PageService }
from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [PageService],
  standalone: true,
  selector: 'app-container',
  encapsulation: ViewEncapsulation.None,
  template: `<ejs-treegrid #treegrid [dataSource]='data' height=230
[treeColumnIndex]='1' [allowPaging]='true' [pageSettings]='pageSettings'
(dataBound)='dataBound()' childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pageSettings?: Object;
  public toolbar?: ToolbarItems[];
  public initialGridLoad: boolean = true;
  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;

  ngOnInit(): void {
```

```

        this.data = sampleData;
        this.pageSettings = { pageSizes: true, pageSize: 12 };
    }
    dataBound() {
        if (this.initialGridLoad) {
            this.initialGridLoad = false;
            const pager = document.getElementsByClassName('e-gridpager');
            let topElement;
            if ((this.treegrid as any).toolbar) {
                topElement = document.getElementsByClassName('e-toolbar');
            } else {
                topElement = document.getElementsByClassName('e-
gridheader');
            }
            topElement[0].before(pager[0]);
        }
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
                priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 6,
        taskName: 'Design',

```

```

        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
              endDate: new Date('02/12/2017'), duration: 3, progress: 60,
              priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
              endDate: new Date('02/12/2017'), duration: 3, progress: 100,
              priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
              endDate: new Date('02/14/2017'), duration: 2, progress: 100,
              priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
              endDate: new Date('02/14/2017'), duration: 2, progress: 100,
              priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
              endDate: new Date('02/14/2017'), duration: 0, progress: 0,
              priority: 'Normal', approved: true }
        ]
    },
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,
                    taskName: 'Implementation Module 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'Normal',
                    duration: 11,
                    progress: 10,
                    approved: false,

```

```

        subtasks: [
            { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
            { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
            { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
                { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
                    endDate: new Date('02/21/2017'), duration: 0,
progress: '0', priority: 'Normal', approved: false }
            ]
        }
    ]
}

```

```

        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  }
},
{
  taskID: 29,
  taskName: 'Phase 3',
  startDate: new Date('02/17/2017'),
  endDate: new Date('02/27/2017'),
  priority: 'Normal',
  approved: false,
  duration: 11,
  progress: 30,
  subtasks: [{
    taskID: 30,
    taskName: 'Implementation Module 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'High',
    approved: false,
    duration: 11,
    progress: 60,
    subtasks: [
      { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
      { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
      { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
      { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
      { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
    ]
  }
]
}
}

```

```

        { taskID: 36, taskName: 'Phase 3 complete',
        startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
        progress: '50', priority: 'Critical', approved: false },
      ]
    }
  ]
}
];

```

MAIN.TS

```

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

```

During the paging action, the pager component triggers the below three events.

- * The [created](#) event triggers when Pager is created.
- * The [click](#) event triggers when the numeric items in the pager is clicked.
- * The [dropDownChanged](#) event triggers when pageSize DropDownList value is selected.

Pager events

The TreeGrid component triggers two pager events during paging actions:

[actionBegin](#)- This event triggered before any paging action (such as changing the page, changing the page size and etc) is initiated. You can use this event to customize or control the behavior of paging actions.

[actionComplete](#)- This event triggered after a pager action is completed. It provides information about the action, such as the new page number, page size, and the total number of records. You can use this event to perform actions or update the UI after the operation has been executed.

The following example demonstrates how to use these events to display notification messages to indicate the current and next page during paging actions in the tree grid:

APP.COMPONENT.TS

```

{% raw %}
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild, ViewEncapsulation } from
 '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, ToolbarItems, PageSettingsModel, PageService }
from '@syncfusion/ej2-angular-treegrid';
import { PageEventArgs } from '@syncfusion/ej2-grids';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers:[PageService],

```

```

standalone: true,
selector: 'app-container',
encapsulation: ViewEncapsulation.None,
template: ` <p id="message"
style="color:red;align:center">{{actioncomplete_message}}</p>
<p id="message1" style="color:red;align:center">{{actionbegin_message}}</p>
<ejs-treegrid #treegrid [dataSource]='data' height=230
[treeColumnIndex]='1' [allowPaging]='true' [pageSettings]='pageSettings'
(actionBegin)="onActionBegin($event)"
(actionComplete)="onActionComplete($event)" childMapping='subtasks' >
<e-columns>
<e-column field='taskID' headerText='Task ID' textAlign='Right'
width=90></e-column>
<e-column field='taskName' headerText='Task Name' textAlign='Left'
width=180></e-column>
<e-column field='startDate' headerText='Start Date' textAlign='Right'
format='yMd' width=90></e-column>
<e-column field='duration' headerText='Duration' textAlign='Right'
width=80></e-column>
</e-columns>
</ejs-treegrid>`
))
export class AppComponent implements OnInit {
public data?: Object[];
public pageSettings?: Object ;
public actioncomplete_message?: string;
public actionbegin_message?: string;
@ViewChild('treegrid')
public treegrid?: TreeGridComponent;
ngOnInit(): void {
this.data = sampleData;
this.pageSettings = { pageSize: 8 };
}
onActionBegin({requestType,currentPage,previousPage}: PageEventArgs) {
if (requestType === 'paging') {
this.actionbegin_message = (currentPage as string) > (previousPage as
string)
? `You are going to switch to page ${parseInt((currentPage as string), 10) +
1}`
: `You are going to switch to page ${previousPage}`;
}
}
onActionComplete(args: PageEventArgs) {
if (args.requestType === 'paging') {
this.actioncomplete_message= 'Now you are in page ' + args.currentPage;
}
}
}
{% endraw %}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [

```



```

{
  taskID: 1,
  taskName: 'Planning',
  startDate: new Date('02/03/2017'),
  endDate: new Date('02/07/2017'),
  progress: 100,
  duration: 5,
  priority: 'Normal',
  approved: false,
  subtasks: [
    { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
      endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
    { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
      endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
    { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
      endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
    { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
      endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
  ],
},
{
  taskID: 6,
  taskName: 'Design',
  startDate: new Date('02/10/2017'),
  endDate: new Date('02/14/2017'),
  duration: 3,
  progress: 86,
  priority: 'High',
  approved: false,
  subtasks: [
    { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
      endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
    { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
      endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
    { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
      endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
    { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
      endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
    { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),

```

```

        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
    ],
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,
                    taskName: 'Implementation Module 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'Normal',
                    duration: 11,
                    progress: 10,
                    approved: false,
                    subtasks: [
                        { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
                        { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                        { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
                        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
                            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
                    ]
                }
            ]
        }
    ]
}

```

```

    ]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [
            {
                taskID: 22,
                taskName: 'Implementation Module 2',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/28/2017'),
                priority: 'Critical',
                approved: false,
                duration: 12,
                progress: 90,
                subtasks: [
                    { taskID: 23, taskName: 'Development Task 1',
                        startDate: new Date('02/17/2017'),
                        endDate: new Date('02/20/2017'), duration: 4,
                        progress: '50', priority: 'Normal', approved: true },
                    { taskID: 24, taskName: 'Development Task 2',
                        startDate: new Date('02/17/2017'),
                        endDate: new Date('02/20/2017'), duration: 4,
                        progress: '50', priority: 'Critical', approved: true },
                    { taskID: 25, taskName: 'Testing', startDate: new
                        Date('02/21/2017'),
                        endDate: new Date('02/24/2017'), duration: 2,
                        progress: '0', priority: 'High', approved: false },
                    { taskID: 26, taskName: 'Bug fix', startDate: new
                        Date('02/25/2017'),
                        endDate: new Date('02/26/2017'), duration: 2,
                        progress: '0', priority: 'Low', approved: false },
                    { taskID: 27, taskName: 'Customer review meeting',
                        startDate: new Date('02/27/2017'),
                        endDate: new Date('02/28/2017'), duration: 2,
                        progress: '0', priority: 'Critical', approved: true },
                    { taskID: 28, taskName: 'Phase 2 complete',
                        startDate: new Date('02/28/2017'),
                        endDate: new Date('02/28/2017'), duration: 0,
                        progress: '50', priority: 'Normal', approved: false }
                ]
            }
        ]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
    }
]

```

```

        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'Normal', approved: false },
                { taskID: 33, taskName: 'Testing', startDate: new
                    Date('02/20/2017'),
                    endDate: new Date('02/21/2017'), duration: 2,
                    progress: '0', priority: 'Critical', approved: true },
                { taskID: 34, taskName: 'Bug fix', startDate: new
                    Date('02/24/2017'),
                    endDate: new Date('02/25/2017'), duration: 2,
                    progress: '0', priority: 'High', approved: false },
                { taskID: 35, taskName: 'Customer review meeting',
                    startDate: new Date('02/26/2017'),
                    endDate: new Date('02/27/2017'), duration: 2,
                    progress: '0', priority: 'Normal', approved: true },
                { taskID: 36, taskName: 'Phase 3 complete',
                    startDate: new Date('02/27/2017'),
                    endDate: new Date('02/27/2017'), duration: 0,
                    progress: '50', priority: 'Critical', approved: false },
            ]
        }]
    }
]
};

```

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 change loading indicator in Angular TreeGrid](#)

Scrolling in Angular Treegrid component

The scrollbar will be displayed in the treegrid when content exceeds the element [width](#) or [height](#). The vertical and horizontal scrollbars will be displayed based on the following criteria:

- The vertical scrollbar appears when the total height of rows present in the treegrid exceeds its element height.
- The horizontal scrollbar appears when the sum of columns width exceeds the treegrid element width.
- The [height](#) and [width](#) are used to set the treegrid height and width, respectively.

The default value for [height](#) and [width](#) is auto.

Set width and height

To specify the [width](#) and [height](#) of the scroller in the pixel, set the pixel value to a number.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit } from '@angular/core';
import { sampleData } from '../datasource';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' height='250' width='400'
[treeColumnIndex]='1' childMapping='subtasks' >
  <e-columns>
    <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
    <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
    <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
    <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
  </e-columns>
</ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  ngOnInit(): void {
    this.data = sampleData;
  }
}
```

```
}

```

MAIN.TS

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

Responsive with parent container

Specify the [width](#) and [height](#) as **100%** to make the treegrid element fill its parent container.

Setting the [height](#) to **100%** requires the treegrid parent element to have explicit height.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `
    <div style="height: 350px">
      <ejs-treegrid [dataSource]='data' height='100%' width='100%'
[treeColumnIndex]='1' childMapping='subtasks' >
        <e-columns>
          <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
          <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
          <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
          <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
        </e-columns>
      </ejs-treegrid>
    </div>
  `
})
export class AppComponent implements OnInit {
  public data?: Object[];
  ngOnInit(): void {
```

```

        this.data = sampleData;
    }
}

```

MAIN.TS

```

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

```

Scroll to selected row

Scroll the tree grid content to the selected row position by using the [rowSelected](#) event.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { NumericTextBoxModule } from '@syncfusion/ej2-angular-inputs'
import { RowSelectEventArgs } from '@syncfusion/ej2-grids';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { NumericTextBoxComponent } from '@syncfusion/ej2-angular-inputs';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    NumericTextBoxModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-numerictextbox #numerictext min="0" max="50" width='200'
    [showSpinButton]='false' format= 'N' value="0" placeholder='Enter index to
    select a row' (change)='onChange($event)' ></ejs-numerictextbox>
    <ejs-treegrid #treegrid [dataSource]='data' height='260' width='100%'
    [treeColumnIndex]='1' childMapping='subtasks'
    (rowSelected)='rowSelected($event)' [selectedIndex]='0'>
      <e-columns>
        <e-column field='taskId' headerText='Task ID'
        textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
        textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
        textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
        textAlign='Right' width=110></e-column>
      </e-columns>
    </ejs-treegrid>`
  })
export class AppComponent implements OnInit {

```

```

public data?: Object[];
ngOnInit(): void {
    this.data = sampleData;
}
@ViewChild('treegrid')
public treeGridObj?: TreeGridComponent;
@ViewChild('numericText')
public numeric?: NumericTextBoxComponent;
onChange(args: any): void {
    this.treeGridObj?.selectRow(parseInt((this.numeric as
NumericTextBoxComponent).getText(), 10));
}
rowSelected(args: RowSelectEventArgs) {
    let rowHeight: number = (this.treeGridObj as
TreeGridComponent).getRows()[this.treeGridObj as
TreeGridComponent].getSelectedRowIndex()[0].scrollHeight;
    (this.treeGridObj as
TreeGridComponent).getContent().children[0].scrollTop = rowHeight *
(this.treeGridObj as TreeGridComponent).getSelectedRowIndex()[0];
}
}

```

MAIN.TS

```

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

```

Hide the scrollbar when the content is not overflown

Hide the scrollbar of the tree grid content by using the [hideScroll](#) method when the content doesn't overflow its parent element.

In the following sample, the [hideScroll](#) method is invoked inside the [dataBound](#) event.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { NumericTextBoxModule } from '@syncfusion/ej2-angular-inputs'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
    imports: [
        TreeGridModule,
        ButtonModule,
        NumericTextBoxModule
    ],
    standalone: true,
    selector: 'app-container',

```



```

    template: `<ejs-treegrid #treegrid [dataSource]='data' height='250'
    [treeColumnIndex]='1' childMapping='subtasks'
    (dataBound)='dataBound($event)'>
        <e-columns>
            <e-column field='taskID' headerText='Task ID'
            textAlign='Right' width=90></e-column>
            <e-column field='taskName' headerText='Task Name'
            textAlign='Left' width=180></e-column>
            <e-column field='startDate' headerText='Start Date'
            textAlign='Right' format='yMd' width=120></e-column>
            <e-column field='duration' headerText='Duration'
            textAlign='Right' width=110></e-column>
        </e-columns>
    </ejs-treegrid>`
  })
  export class AppComponent implements OnInit {
    public data?: object[];
    @ViewChild('treegrid')
    public treegrid?: TreeGridComponent;
    ngOnInit(): void {
      this.data = sampleData.slice(0, 1);
    }
    dataBound(args: any): void {
      this.treegrid?.grid.hideScroll();
    }
  }

```

MAIN.TS

```

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

```

Frozen in Angular Treegrid component

Frozen rows and columns

Frozen rows and columns provides an option to make rows and columns always visible in the top and left side of the tree grid while scrolling.

In this demo, the [frozenColumns](#) is set as '2' and the [frozenRows](#) is set as '3'. Hence, the left two columns and top three rows are frozen.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { NumericTextBoxModule } from '@syncfusion/ej2-angular-inputs'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
import { Component, OnInit, ViewEncapsulation, ViewChild } from
'@angular/core';
import { FreezeService, TreeGridComponent } from '@syncfusion/ej2-angular-
treegrid';
import { NumericTextBoxComponent } from '@syncfusion/ej2-angular-inputs';
import { sampleData } from './datasource';

```

```

@Component({
  imports: [

    TreeGridModule, NumericTextBoxModule, ButtonModule

  ],
  standalone: true,
  selector: 'app-container',
  template: ` <div style="display: flex">
    <label style="padding: 10px 10px 26px 0">Change the
    frozen columns:</label>

    <ejs-numerictextbox id="frozenColumns" #frozenColumns
    min="0" max="5" [validateDecimalOnType]="true" decimals="0" format="n"
    value="2" width="100px" ></ejs-numerictextbox>
    <div>
      <button style="margin-left:5px" ejs-button
    (click)="frozenColumnFn()">Update</button>
    </div>
    </div>
    <ejs-treegrid #treegrid [dataSource]='data'
    childMapping='subtasks' [treeColumnIndex]='1' height='310'
    [frozenColumns]='2' [frozenRows]='3' [allowSelection]='false'>
      <e-columns>
        <e-column field='taskID' headerText='Task ID' width='90'
        textAlign='Right'></e-column>
        <e-column field='taskName' headerText='Task Name'
        width='230'></e-column>
        <e-column field='startDate' headerText='Start Date'
        width='120' format='yMd' textAlign='Right'></e-column>
        <e-column field='endDate' headerText='End Date'
        width='120' format='yMd' textAlign='Right'></e-column>
        <e-column field='duration' headerText='Duration'
        width='110' textAlign='Right'></e-column>
        <e-column field='progress' headerText='Progress'
        width='110' textAlign='Right'></e-column>
        <e-column field='priority' headerText='Priority'
        width='110'></e-column>
        <e-column field='approved' headerText='Approved'
        textAlign='Left' width='110'></e-column>
      </e-columns>
    </ejs-treegrid>`,
  providers: [FreezeService]
})
export class AppComponent implements OnInit {
  public data?: Object[];

  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  @ViewChild('frozenColumns')
  public frozenColumns?: NumericTextBoxComponent;
  ngOnInit(): void {
    this.data = sampleData;
  }
  frozenColumnFn() {
    (this.treegrid as TreeGridComponent).frozenColumns =
    (this.frozenColumns as NumericTextBoxComponent).value;
  }
}

```

```
}
}
```

MAIN.TS

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

Freeze particular columns

You can use [isFrozen](#) property to freeze selected columns in tree grid.

In this demo, the columns with field name `taskName` and `startDate` is frozen using the `isFrozen` property.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule, ToolbarService, SelectionService, EditService }
from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit, ViewEncapsulation, ViewChild } from
'@angular/core';
import { FreezeService, TreeGridComponent } from '@syncfusion/ej2-angular-
treegrid';
import { sampleData } from './datasource';
@Component({
  imports: [
    TreeGridModule
  ],
  providers: [ToolbarService,
    SelectionService,
    EditService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' childMapping='subtasks'
height='310' [allowSelection]='false'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
      <e-column field='taskName' headerText='Task Name' width='200'
isFrozen= 'true'></e-column>
      <e-column field='startDate' headerText='Start Date' isFrozen=
'true' width='150' format='yMd' textAlign='Right'></e-column>
      <e-column field='endDate' headerText='End Date' width='150'
format='yMd' textAlign='Right'></e-column>
      <e-column field='duration' headerText='Duration' width='110'
textAlign='Right'></e-column>
      <e-column field='progress' headerText='Progress' width='110'
textAlign='Right'></e-column>
      <e-column field='priority' headerText='Priority'
width='110'></e-column>
```

```

        <e-column field='approved' headerText='Approved'
textAlign='Left' width='110'></e-column>
    </e-columns>
</ejs-treegrid>`,
providers: [FreezeService]
})
export class AppComponent implements OnInit {
    public data?: Object[];
    ngOnInit(): void {
        this.data = sampleData;
    }
}

```

MAIN.TS

```

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

```

Freeze direction

You can freeze the tree grid columns on the left or right side by using the [column.freeze](#) property and the remaining columns will be movable. The grid will automatically move the columns to the left or right position based on the [column.freeze](#) value.

Types of the [column.freeze](#) directions:

- **Left:** Allows you to freeze the columns at the left.
- **Right:** Allows you to freeze the columns at the right.

In this demo, the **Task Name** column is frozen at the left and the **Priority** column is frozen at the right side of the content table.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule, ToolbarService, SelectionService, EditService }
from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit, ViewEncapsulation, ViewChild } from
'@angular/core';
import { FreezeService, TreeGridComponent } from '@syncfusion/ej2-angular-
treegrid';
import { sampleData } from './datasource';
@Component({
imports: [

    TreeGridModule
],
providers: [ToolbarService,
    SelectionService,
    EditService],
standalone: true,
selector: 'app-container',

```

```

    template: `<ejs-treegrid [dataSource]='data' childMapping='subtasks'
height='310' [treeColumnIndex]='1' [allowSelection]='false'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID' width='90'
textAlign='Right'></e-column>
        <e-column field='taskName' headerText='Task Name' width='200'
freeze='Left'></e-column>
        <e-column field='startDate' headerText='Start Date' width='150'
format='yMd' textAlign='Right'></e-column>
        <e-column field='endDate' headerText='End Date' width='150'
format='yMd' textAlign='Right'></e-column>
        <e-column field='duration' headerText='Duration' width='110'
textAlign='Right'></e-column>
        <e-column field='progress' headerText='Progress' width='110'
textAlign='Right'></e-column>
        <e-column field='priority' headerText='Priority' width='110'
freeze='Right'></e-column>
        <e-column field='approved' headerText='Approved'
textAlign='Left' width='110'></e-column>
    </e-columns>
    </ejs-treegrid>`,
providers: [FreezeService]
})
export class AppComponent implements OnInit {
    public data?: Object[];
    ngOnInit(): void {
        this.data = sampleData;
    }
}

```

MAIN.TS

```

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

```

- * Freeze Direction is not compatible with the [isFrozen](#) and [frozenColumns](#) properties.
- * You can get the frozen right rows by using the [getFrozenRightRows](#) method in tree grid.
- * To get the movable rows, you can use the [getMovableRows](#) method in tree grid.
- * You can get the movable row by index using the [getMovableRowByIndex](#) method in tree grid.
- * To get the movable cell from index using the [getMovableCellFromIndex](#) method in tree grid. In this method, you need to pass the row index and column index of movable column as parameter.
- * To get the frozen right column header element by index, you can use the [getFrozenRightColumnHeaderByIndex](#) method in tree grid. In this method, you need to pass the frozen right column header index as parameter.
- * To get the frozen left column header element by index, you can use the [getFrozenLeftColumnHeaderByIndex](#) method in tree grid. In this method, you need to pass the frozen left column header index as parameter.

* To get the frozen right cell element from index, you can use the [getFrozenRightCellFromIndex](#) method in tree grid. In this method, you need to pass the row index and cell index of the frozen right column as parameter.

To get the movable column header element by index, you can use the [getMovableColumnHeaderByIndex](#) method in tree grid. In this method, you need to pass the movable column index as parameter.

* To get the frozen right row element by index, you can use the [getFrozenRightRowByIndex](#) method in tree grid. In this method, you need to pass the right frozen row index as parameter.

Limitations of frozen tree grid

The following features are not supported in frozen rows and columns:

- Row Template
- Detail Template
- Cell Editing

Limitations of freeze direction

This feature has the following limitations, along with the above mentioned frozen tree grid limitations.

- Infinite scroll cache mode.
- Freeze direction in the stacked header is not compatible with column reordering.

Add validation rule for frozen tree grid

In a frozen column-enabled tree grid, the content will be separated into frozen and movable parts. The following code is used to dynamically add validation to input fields in the movable part. In the [actionComplete](#) event args, find the movableform instance as an argument. Here, add the validation rules dynamically.

`typescript

```
actionComplete: (args: DialogEditEventArgs) => {
  if ((args.requestType === 'beginEdit' || args.requestType === 'add')) {
    // Add Validation Rules
    args.movableForm.ej2_instances[0].addRules('duration', { max: 200 }); // Here, 'duration' is the column
    name.
  }
}
```

Validation rules for the 'duration' and 'taskId' columns can be added in the following sample.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule, ToolbarService, SelectionService, EditService }
  from '@syncfusion/ej2-angular-treegrid'
```

```

import { TreeGrid, Selection, Edit, Toolbar, EditSettingsModel,
FreezeService } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
@Component({
imports: [

    TreeGridModule
],
providers: [ToolbarService,
    SelectionService,
    EditService],
standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid [dataSource]='data' childMapping='subtasks'
height='310' [treeColumnIndex]='0' [toolbar]='toolbar'
[editSettings]='editSettings' allowSelection='false' enableHover='false'
(actionComplete)="actionComplete($event)">
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
[validationRules]='validationrules' width='90' textAlign='Right'></e-column>
        <e-column field='taskName' headerText='Task Name' width='200'
freeze='Left'></e-column>
        <e-column field='duration' headerText='Duration'
[validationRules]='validationrules' width='110' textAlign='Right'></e-
column>
        <e-column field='startDate' headerText='Start Date' width='150'
format='yMd' textAlign='Right'></e-column>
        <e-column field='endDate' headerText='End Date' width='150'
format='yMd' textAlign='Right'></e-column>
        <e-column field='progress' headerText='Progress' width='110'
textAlign='Right'></e-column>
        <e-column field='priority' headerText='Priority' width='110'
freeze='Right'></e-column>
        <e-column field='approved' headerText='Approved'
textAlign='Left' width='110'></e-column>
    </e-columns>
    </ejs-treegrid>`,
    providers: [FreezeService]
})
export class AppComponent implements OnInit {
    public data?: object[];
    public toolbar?: string[];
    public editSettings?: EditSettingsModel;
    public validationrules?: Object;
    ngOnInit(): void {
        this.data = sampleData;
        this.toolbar = ['Add', 'Edit', 'Delete', 'Update', 'Cancel'];
        this.editSettings = { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Row' };
        this.validationrules = { required: true };
    }
    actionComplete(args: any) {
        if (args.requestType === 'beginEdit' || args.requestType === 'add')
        {
            // Add Validation Rules

```

```

    args.movableForm.ej2_instances[0].addRules('duration', { max:
200 });
    args.movableForm.ej2_instances[0].addRules('taskID', { max: 20
});
    }
  }
}

```

MAIN.TS

```

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

```

* This is applicable when a frozen column is enabled and the edit mode is set as "**Row**" in the tree grid.

* You can refer to Syncfusion [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. Also, explore Syncfusion [Angular Tree Grid example](#) to know how to present and manipulate data.

Virtual scroll in Angular Treegrid component

TreeGrid allows you to load large amount of data without performance degradation.

To use virtualization, you need to inject `VirtualScrollService` in TreeGrid.

Row virtualization

Row virtualization allows you to load and render rows only in the content viewport. It is an alternative way of paging in which the rows will be appended while scrolling vertically. To setup the row virtualization, you need to define `enableVirtualization` as true and content height by `height` property.

The number of records displayed in the TreeGrid is determined implicitly by height of the content area and a buffer records will be maintained in the TreeGrid content in addition to the original set of rows.

Expand and Collapse state of any child record will be persisted.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit, ViewEncapsulation, ViewChild } from
'@angular/core';
import { VirtualScrollService, TreeGridComponent } from '@syncfusion/ej2-
angular-treegrid';
import { DataSource, virtualData } from './datasource';
@Component({
  imports: [
    TreeGridModule
  ],
  standalone: true,
  selector: 'app-container',

```



```

    template: `<ejs-treegrid #treegrid [dataSource]='data'
[enableVirtualization]=true height=291 childMapping='Crew'
[treeColumnIndex]='1' >
      <e-columns>
        <e-column field='TaskID' headerText='Player Jersey' width='120'
textAlign='Right'></e-column>
        <e-column field='FIELD1' headerText='Player Name'
width='120'></e-column>
        <e-column field='FIELD2' headerText='Year' width='100'
textAlign='Right'></e-column>
        <e-column field='FIELD3' headerText='Stint' width='120'
textAlign='Right'></e-column>
        <e-column field='FIELD4' headerText='TMID' width='120'
textAlign='Right'></e-column>
      </e-columns>
    </ejs-treegrid>`,
    providers: [VirtualScrollService]
  })
  export class AppComponent implements OnInit {
    public data?: Object[];
    ngOnInit(): void {
      dataSource();
      this.data = virtualData;
    }
  }
}

```

MAIN.TS

```

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

```

Column virtualization

Column virtualization allows you to virtualize columns. It will render column only in the current view port and all other columns are rendered on demand during horizontal scrolling.

To setup the column virtualization, set the [enableVirtualization](#) and [enableColumnVirtualization](#) properties as `true`.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit, ViewEncapsulation, ViewChild } from
 '@angular/core';
import { VirtualScrollService, TreeGridComponent } from '@syncfusion/ej2-
angular-treegrid';
import { dataSource, virtualData } from './datasource';
@Component({
  imports: [

    TreeGridModule
  ],

```

```

,
standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data'
[enableVirtualization]=true [enableColumnVirtualization]=true height=291
childMapping='Crew' [treeColumnIndex]='1' >
    <e-columns>
      <e-column field='TaskID' headerText='Player Jersey' width='120'
textAlign='Right'></e-column>
      <e-column field='FIELD1' headerText='Player Name'
width='120'></e-column>
      <e-column field='FIELD2' headerText='Year' width='100'
textAlign='Right'></e-column>
      <e-column field='FIELD3' headerText='Stint' width='120'
textAlign='Right'></e-column>
      <e-column field='FIELD4' headerText='TMID' width='120'
textAlign='Right'></e-column>
      <e-column field='FIELD5' headerText='LGID' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD6' headerText='GP' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD7' headerText='GS' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD8' headerText='Minutes' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD9' headerText='Points' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD10' headerText='oRebounds' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD11' headerText='dRebounds' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD12' headerText='Rebounds' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD13' headerText='Assists' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD14' headerText='Steals' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD15' headerText='Blocks' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD16' headerText='Turnovers' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD17' headerText='PF' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD18' headerText='fgAttempted' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD19' headerText='ftAttempted' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD20' headerText='ThreeAttempted' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD21' headerText='ThreeMade' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD22' headerText='PostGP' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD23' headerText='ftMade' width= 120
textAlign='Right'></e-column>
      <e-column field='FIELD24' headerText='fgMade' width= 120
textAlign='Right'></e-column>

```

```

        <e-column field='FIELD25' headerText='ffmade' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD26' headerText='PostGS' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD27' headerText='PostMinutes' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD28' headerText='PostPoints' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD29' headerText='PostoRebounds' width= 120
textAlign='Right'></e-column>
        <e-column field='FIELD30' headerText='PostdRebounds' width= 120
textAlign='Right'></e-column>
    </e-columns>
</ejs-treegrid>`,
providers: [VirtualScrollService]
})
export class AppComponent implements OnInit {
    public data?: Object[];
    ngOnInit(): void {
        dataSource();
        this.data = virtualData;
    }
}

```

MAIN.TS

```

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

```

Column's [width](#) is required for column virtualization.

If column's **width** is not defined then tree grid will consider its value as **200px**.

Limitations for virtualization

- Due to the element height limitation in browsers, the maximum number of records loaded by the treegrid is limited by the browser capability.
- Cell selection will not be persisted in row.
- Virtual scrolling is not compatible with detail template.
- The page size provided must be two times larger than the number of visible rows in the TreeGrid. If the page size is failed to meet this condition then the size will be determined by TreeGrid.
- The virtual height of the treegrid content is calculated using the row height and total number of records in the data source and hence features which changes row height such as text wrapping are not supported. If you want to increase the row height to accommodate the content then you can specify the row height as below to ensure all the table rows are in same height.

`css

```
.e-treegrid .e-row {
```

```
height: 2em;
```

```
}
```

```
,
```

- Programmatic selection using the **selectRows** method is not supported in virtual scrolling.
- Virtual scrolling is not compatible with Batch editing, clipboard functionality and detail template.
- When virtualization is active in a tree grid, the editCell method is unusable for records outside the currently visible viewport.

You can refer to our [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Infinite scroll in Angular Treegrid component

Infinite scrolling is used to load a huge amount of data without degrading the Tree Grid performance. This feature works like the lazy loading concept, which means the buffer data is loaded only when the scrollbar reaches the end of the scroller.

To use Infinite scrolling, set **enableInfiniteScrolling** property as true and inject the **InfiniteScroll** module in the treegrid.

* In this feature, Tree Grid will not make a new data request when you visit the same page again.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, InfiniteScrollService } from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit } from '@angular/core';
import { dataSource, virtualData } from './datasource';
@Component({
  imports: [

    TreeGridModule
  ],
  providers: [PageService,
    InfiniteScrollService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data'
[enableInfiniteScrolling]=true height=317 [pageSettings]='pageSettings'
childMapping='Crew' [treeColumnIndex]='1' >
    <e-columns>
      <e-column field='TaskID' headerText='Player Jersey' width='120'
textAlign='Right'></e-column>
      <e-column field='FIELD1' headerText='Player Name'
width='120'></e-column>
      <e-column field='FIELD2' headerText='Year' width='100'
textAlign='Right'></e-column>
      <e-column field='FIELD3' headerText='Stint' width='120'
textAlign='Right'></e-column>
      <e-column field='FIELD4' headerText='TMID' width='120'
textAlign='Right'></e-column>
    </e-columns>
  `
})
export class AppComponent implements OnInit {
  ngOnInit() {
    // ...
  }
}
```

```

        </e-columns>
    </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public pageSettings?: Object;
        ngOnInit(): void {
            dataSource();
            this.data = virtualData;
            this.pageSettings = {pageSize: 30};
        }
    }
}

```

MAIN.TS

```

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

```

InitialBlocks

You can define the initial loading pages count by using `infiniteScrollSettings.initialBlocks` property. By default, this feature loads three pages in initial rendering.

In the below demo, we have changed this property value to load five page records instead of three.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, InfiniteScrollService } from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit } from '@angular/core';
import { dataSource, virtualData } from './datasource';
@Component({
    imports: [

        TreeGridModule
    ],
    providers: [PageService,
        InfiniteScrollService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid #treegrid [dataSource]='data'
[enableInfiniteScrolling]=true height=317
[infiniteScrollSettings]='infiniteScrollSettings'
[pageSettings]='pageSettings' childMapping='Crew' [treeColumnIndex]='1' >
        <e-columns>
            <e-column field='TaskID' headerText='Player Jersey' width='120'
textAlign='Right'></e-column>
            <e-column field='FIELD1' headerText='Player Name'
width='120'></e-column>
            <e-column field='FIELD2' headerText='Year' width='100'
textAlign='Right'></e-column>

```

```

        <e-column field='FIELD3' headerText='Stint' width='120'
textAlign='Right'></e-column>
        <e-column field='FIELD4' headerText='TMID' width='120'
textAlign='Right'></e-column>
    </e-columns>
</ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public pageSettings?: Object;
        public infiniteScrollSettings?: Object;
        ngOnInit(): void {
            dataSource();
            this.data = virtualData;
            this.pageSettings = {pageSize: 30};
            this.infiniteScrollSettings = { initialBlocks: 5};
        }
    }
}

```

MAIN.TS

```

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

```

Cache Mode

Cache is used to store the loaded rows object in the Tree Grid instance which can be reused for creating the row elements whenever you scroll to already visited page. Also, this mode maintains row elements based on the `infiniteScrollSettings.maxBlocks` count value, once this limit exceeds then it will remove row elements from DOM for new rows.

To enable the cache mode in Infinite scrolling, set `infiniteScrollSettings.enableCache` property as true.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, InfiniteScrollService } from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit } from '@angular/core';
import { dataSource, virtualData } from './datasource';
@Component({
  imports: [

    TreeGridModule
  ],
  providers: [PageService,
    InfiniteScrollService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data'
[enableInfiniteScrolling]=true height=317

```

```

[infiniteScrollSettings]='infiniteScrollSettings'
[pageSettings]='pageSettings' childMapping='Crew' [treeColumnIndex]='1' >
  <e-columns>
    <e-column field='TaskID' headerText='Player Jersey' width='120'
textAlign='Right'></e-column>
    <e-column field='FIELD1' headerText='Player Name'
width='120'></e-column>
    <e-column field='FIELD2' headerText='Year' width='100'
textAlign='Right'></e-column>
    <e-column field='FIELD3' headerText='Stint' width='120'
textAlign='Right'></e-column>
    <e-column field='FIELD4' headerText='TMID' width='120'
textAlign='Right'></e-column>
  </e-columns>
</ejs-treegrid>`
}))
export class AppComponent implements OnInit {
  public data?: Object[];
  public pageSettings?: Object;
  public infiniteScrollSettings?: Object;
  ngOnInit(): void {
    dataSource();
    this.data = virtualData;
    this.pageSettings = {pageSize: 30};
    this.infiniteScrollSettings = { enableCache: true};
  }
}

```

MAIN.TS

```

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

```

Limitations for Virtualization

- Due to the element height limitation in browsers, the maximum number of records loaded by the tree grid is limited due to the browser capability.
- Initial loading rows total height must be greater than the viewport height.
- Cell selection will not be persisted in cache mode.
- Infinite scrolling is not compatible with batch editing, cell editing, detail template and hierarchy features.
- The aggregated information and total group items are displayed based on the current view items. To get these information regardless of the view items, refer to the
- Programmatic selection using the [selectRows](#) and [selectRow](#) method is not supported in infinite scrolling.

You can refer to our [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Selection

Selection in Angular Treegrid component

Selection provides an option to highlight a row or cell. Selection can be done through simple Mouse down or Arrow keys. To disable selection in the TreeGrid, set the [allowSelection](#) to false.

The treegrid supports two types of selection that can be set by using the [selectionSettings.type](#). They are:

- **Single** - The **Single** value is set by default. Allows you to select only a single row or cell.
- **Multiple** - Allows you to select multiple rows or cells.

To perform the multi-selection, press and hold CTRL key and click the desired rows or cells.

To select range of rows or cells, press and hold the SHIFT key and click the rows or cells.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit } from '@angular/core';
import { sampleData } from '../datasource';
import { SelectionSettingsModel } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
[allowPaging]='true' [selectionSettings]='selectionOptions'
childMapping='subtasks' >
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
```



```

public data?: Object[];
public selectionOptions?: SelectionSettingsModel;
ngOnInit(): void {
    this.data = sampleData;
    this.selectionOptions = { type: 'Multiple' };
}
}

```

MAIN.TS

```

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

```

Selection mode

TreeGrid supports three types of selection mode which can be set by using [selectionSettings.mode](#). They are:

- **Row** - The **row** value is set by default. Allows you to select rows only.
- **Cell** - Allows you to select cells only.
- **Both** - Allows you to select rows and cells at the same time.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
import { SelectionSettingsModel } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule

  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' height='265'
[treeColumnIndex]='1' [allowPaging]='true'
[selectionSettings]='selectionOptions' childMapping='subtasks' >
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>

```

```

        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public selectionOptions?: SelectionSettingsModel;
        ngOnInit(): void {
            this.data = sampleData;
            this.selectionOptions = { mode: 'Both' };
        }
    }
}

```

MAIN.TS

```

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

```

Toggle selection

The Toggle selection allows to perform selection and unselection of the particular row or cell. To [enable toggle](#) selection, set enableToggle property of the selectionSettings as true. If you click on the selected row or cell then it will be unselected and vice versa.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { SelectionSettingsModel } from '@syncfusion/ej2-angular-grids';
@Component({
    imports: [

        TreeGridModule,
        ButtonModule,
        DropDownListAllModule
    ],
    providers: [PageService,
        SortService,
        FilterService],
    standalone: true,
    selector: 'app-container',

```

```

    template: `<ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' childMapping='subtasks'
    height='315px' [selectionSettings]='selectionOptions' >
        <e-columns>
            <e-column field='taskId' headerText='Task ID'
textAlign='Right' width=90></e-column>
            <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
            <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
            <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
        </e-columns>
    </ejs-treegrid>`
  })
  export class AppComponent implements OnInit {
    public data?: Object[];
    public selectionOptions?: SelectionSettingsModel;
    ngOnInit(): void {
      this.data = sampleData;
      this.selectionOptions = { enableToggle: true };
    }
  }
}

```

MAIN.TS

```

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

```

If multi selection is enabled, then first click on any selected row (without pressing Ctrl key), it will clear the multi selection and in second click on the same row, it will be unselected.

Touch interaction

When you tap the tree grid row on touch screen devices, the tapped row is selected.



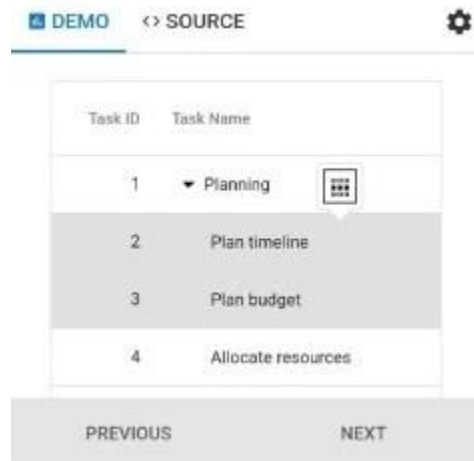
Also, it will show a popup for multi-row-selection.



To select multiple rows or cells, tap the popup then tap the desired rows or cells.

For multi-selection, It requires the selection [type](#) to be **Multiple**.

The following screenshot represents a tree grid touch selection in the device.



Refer to Syncfusion [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. Also, explore Syncfusion [Angular Tree Grid example](#) to know how to present and manipulate data.

Cell selection in Angular Treegrid component

Cell Selection can be done through simple Mouse down or Arrow keys(up, down, left and right).

TreeGrid supports two types of cell selection mode which can be set by using [selectionSettings.cellSelectionMode](#). They are:

- **Flow** - The **Flow** value is set by default.

Select range of cells between the start index and end index which includes in between cells of rows.

- **Box** - Select range of cells within the start and end column indexes which includes in between cells of rows within the range.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { SelectionSettingsModel } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
```

```

standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data' height=250
[treeColumnIndex]='1' [allowPaging]='true' childMapping='subtasks'
[selectionSettings]='selectionOptions'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public selectionOptions?: SelectionSettingsModel;
  ngOnInit(): void {
    this.data = sampleData;
    this.selectionOptions = { cellSelectionMode: 'Box', type:
'Multiple', mode: 'Cell' };
  }
}

```

MAIN.TS

```

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

```

Cell Selection requires the [selectionSettings.mode](#) to be Cell or Both.

Perform cell selection programmatically

To perform cell selection programmatically, you can use [selectCell](#) method. To use this method you need to pass the cellIndex as parameter like in the below sample.

To clear the selected rows or cells, by calling the [clearSelection](#) method in tree grid.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { SelectionSettingsModel } from '@syncfusion/ej2-angular-grids';
import { EditSettingsModel, TreeGridComponent } from '@syncfusion/ej2-
angular-treegrid'

```

```

@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<button ej-button id='selectCell'
(click)='selectCell()'>Select Cell</button>
<button ej-button id='clear' (click)='clear()'>Clear selection</button>
<ejs-treegrid #treegrid [dataSource]='data' height=250
[treeColumnIndex]='1' [editSettings]='editSettings' [allowPaging]='true'
childMapping='subtasks' [selectionSettings]='selectionOptions'>
  <e-columns>
    <e-column field='taskID' headerText='Task ID' textAlign='Right'
width=90></e-column>
    <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
    <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
    <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
  </e-columns>
</ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public editSettings?: EditSettingsModel;
  public data: Object[] = [];
  public selectionOptions?: SelectionSettingsModel;
  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
    this.editSettings = { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: "Cell" };
    this.selectionOptions = { cellSelectionMode: 'Flow', type:
'Multiple', mode: 'Cell' };
  }
  selectCell() {
    this.treegrid?.selectCell({ rowIndex: 3, cellIndex: 1 });
  }
  clear() {
    this.treegrid?.clearSelection();
  }
}

```

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 get selected row cell in tree grid

To get the selected row cell index, Use [getSelectedRowCellIndexes](#) method in the treegrid.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, SelectionSettingsModel } from '@syncfusion/ej2-
angular-treegrid'
import { CellSelectEventArgs } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data'
(cellSelected)='cellSelected($event)' height=250 [treeColumnIndex]='1'
[allowPaging]='true' childMapping='subtasks'
[selectionSettings]='selectionOptions'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID' textAlign='Right'
width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data: Object[] = [];
  public selectionOptions?: SelectionSettingsModel;
  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
    this.selectionOptions = { cellSelectionMode: 'Flow', type:
'Multiple', mode: 'Cell' };
  }
}
```

```

    }
    cellSelected(args: CellSelectEventArgs) {
        var cellSelected: object[] =
        this.treegrid?.getSelectedRowCellIndexes() as object[];
        var cellSelectedCount: number = cellSelected.length;
        alert('Selected row cell count : ' + cellSelectedCount)
    }
}

```

MAIN.TS

```

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

```

Check box selection in Angular Treegrid component

Checkbox Selection provides an option to select multiple TreeGrid records with help of checkbox in each row.

To render checkbox in each treegrid row, you need to use checkbox column with type as **CheckBox** using

column **type** property.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='2' childMapping='subtasks'
height='315px' >
    <e-columns>
        <e-column type='checkbox' width='50'></e-column>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>

```



```

        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        ngOnInit(): void {
            this.data = sampleData;
        }
    }
}

```

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 selection is allowed by clicking a treegrid row or checkbox in that row. To allow Selection only through checkbox, you can set [selectionSettings.checkboxOnly](#) property to true.

Selection can be persisted on all the operations using [selectionSettings.persistSelection](#) property.

For persisting selection on the TreeGrid, any one of the column should be defined as a primary key using [columns.isPrimaryKey](#) property.

Checkbox selection mode

In checkbox selection, selection can also be done by clicking on rows. This selection provides two types of Checkbox Selection mode which can be set by using the following API, [selectionSettings.checkboxMode](#). The modes are;

- **Default:** This is the default value of the checkboxMode. In this mode, user can select multiple rows by clicking rows one by one.
- **ResetOnRowClick:** In ResetOnRowClick mode, when user clicks on a row it will reset previously selected row. Also you can perform multiple-selection in this mode by press and hold CTRL key and click the desired rows. To select range of rows, press and hold the SHIFT key and click the rows.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';

```

```

import { SelectionSettingsModel } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='2' childMapping='subtasks'
height='315px' [selectionSettings]='selectionOptions' >
  <e-columns>
    <e-column type='checkbox' width='50'></e-column>
    <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
    <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
    <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
    <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
  </e-columns>
</ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public selectionOptions?: SelectionSettingsModel;
  ngOnInit(): void {
    this.data = sampleData;
    this.selectionOptions = { checkboxMode: 'ResetOnRowClick' };
  }
}

```

MAIN.TS

```

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

```

Checkbox Selection feature is intended for row selection only; it is not compatible with cell selection mode.

Row selection in Angular Treegrid component

[Select row at initial rendering](#)

To select a row at initial rendering, set the [selectedRowIndex](#) value.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'

```

```

import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
import { SelectionSettingsModel } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [selectedRowIndex]=1
[treeColumnIndex]='1' [selectionSettings]='selectionOptions'
childMapping='subtasks' >
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public selectionOptions?: SelectionSettingsModel;
  ngOnInit(): void {
    this.data = sampleData;
    this.selectionOptions = { type: 'Multiple' };
  }
}

```

MAIN.TS

```

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

```

Get selected row indexes

Selected row indexes can be obtained by using the [getSelectedRowIndex](#) method.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, } from '@syncfusion/ej2-angular-treegrid';
import { RowSelectEventArgs } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' (rowSelected)='rowSelected($event)'
childMapping='subtasks' >
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: object[];
  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
  }
  rowSelected(args: RowSelectEventArgs) {
    const selectedrowindex: number[] = (this.treegrid as
TreeGridComponent).getSelectedRowIndexes(); // Get the selected row
indexes.
    alert(selectedrowindex); // To alert the selected row indexes.
    const selectedrecords: object[] = (this.treegrid as
TreeGridComponent).getSelectedRecords(); // Get the selected records.
  }
}

```

MAIN.TS

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

Clear selection programmatically

Clear the tree grid selection programmatically by using the [clearSelection](#) method.

In the demo below, we initially selected the third row using [selectedRowIndex](#). You can clear this selection by calling the `clearSelection` method in an external button click.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { SelectionSettingsModel, TreeGridComponent } from '@syncfusion/ej2-
angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule

  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<button ej-button class='e-flat' (click)='click()'>Clear
Selection</button>
    <ejs-treegrid #treegrid [selectedRowIndex]='2'
[dataSource]='data' [treeColumnIndex]='1'
[selectionSettings]='selectionOptions' childMapping='subtasks' >
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
      </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
```

```

public data?: Object[];
public selectionOptions?: SelectionSettingsModel;
@ViewChild('treegrid')
public treegrid?: TreeGridComponent;
ngOnInit(): void {
    this.data = sampleData;
    this.selectionOptions = { type: 'Multiple' };
}
click(): void{
    (this.treegrid as TreeGridComponent).clearSelection();
}
}

```

MAIN.TS

```

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

```

Get selected records on various pages

Enabling the [selectionSettings.persistSelection](#) property will persist the selection in all tree grid operations.

So the selection will be maintained on every page even after navigating to another page.

You can get the selected records using the [getSelectedRecords](#) method.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, SelectionSettingsModel, PageSettingsModel } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template:
    `<button ej-button class='e-flat' (click)='click()'>Selected
Records</button>

```

```

    <ejs-treegrid #treegrid [dataSource]='data' allowPaging=true
    [treeColumnIndex]='2' [selectionSettings]='selectionOptions'
    childMapping='subtasks' [pageSettings]='pageOptions'>
        <e-columns>
            <e-column type='checkbox' width=50></e-column>
            <e-column field='taskID' headerText='Task ID'
isPrimaryKey='true' textAlign='Right' width=90></e-column>
            <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
            <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
            <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
        </e-columns>
    </ejs-treegrid>`
  })
  export class AppComponent implements OnInit {
    public data?: object[];
    public selectionOptions?: SelectionSettingsModel;
    public pageOptions?: PageSettingsModel;
    @ViewChild('treegrid')
    public treegrid?: TreeGridComponent;
    ngOnInit(): void {
      this.data = sampleData;
      this.selectionOptions = { type: 'Multiple', persistSelection: true
    };
      this.pageOptions = { pageSize: 5 };
    }
    click(): void{
      let selectedrecords: Object[] = (this.treegrid as
TreeGridComponent).getSelectedRecords(); // get the selected records.
      let selectedRecordsCount: number = selectedrecords.length
      alert(selectedRecordsCount); // to alert the selected records count.
    }
  }
}

```

MAIN.TS

```

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

```

To persist the tree grid selection, it is necessary to define any one of the columns as a primary key using the [columns.isPrimaryKey](#) property.

Get selected rows programmatically

Selected rows can be obtained by using the [getSelectedRows](#) method in the tree grid.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'

```

```

import {ButtonModule} from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, SelectionSettingsModel } from '@syncfusion/ej2-angular-treegrid'
import { RowSelectEventArgs } from '@syncfusion/ej2-angular-grids';
import { count } from 'console';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data'
(rowSelected)="rowSelected($event)" height=250 [treeColumnIndex]='1'
[allowPaging]='true' childMapping='subtasks'
[selectionSettings]='selectionOptions'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID' textAlign='Right'
width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data: Object[] = [];
  public selectionOptions?: SelectionSettingsModel;
  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
    this.selectionOptions = { cellSelectionMode: 'Flow', type:
'Multiple' };
  }
  rowSelected(args: RowSelectEventArgs) {
    debugger
    var selectedRows: Element[] = (this.treegrid as
TreeGridComponent).getSelectedRows();
    var selectedRowsCount: number = selectedRows.length;
    alert('Selected rows count : ' + selectedRowsCount)
  }
}

```

MAIN.TS


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

Aggregates

Aggregates in Angular Treegrid component

Aggregate values are displayed in the TreeGrid footer and in parent row footer for child row aggregate values. It can be configured through `aggregates` property. [field](#) and [type](#) are the minimum properties required to represent an aggregate column.

To use the aggregate feature, you have to inject the `Aggregate` module.

By default, the aggregate value can be displayed in the treegrid footer, and footer of child rows. To show the aggregate value in one of the cells, use the [Link to the Video](#).

You can also check this video to learn about how to use Aggregates in Angular TreeGrid.

Built-in aggregate types

The aggregate type should be specified in the [type](#) property to configure an aggregate column.

The built-in aggregates are,

- Sum
- Average
- Min
- Max
- Count
- Truecount
- Falsecount

* Multiple aggregates can be used for an aggregate column by setting the [type](#) property with an array of aggregate types.

* Multiple types for a column is supported only when one of the aggregate templates is used.

Child aggregate

Aggregate value is calculated for child rows, and it is displayed in the parent row footer. Use the [childSummary](#) property to render the child rows aggregate value.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { AggregateService } from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit } from '@angular/core';
import { summaryRowData } from './datasource';
@Component({
  imports: [
    TreeGridModule
  ],
  providers: [AggregateService ],
```

```

standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' height='240'
[treeColumnIndex]='0' childMapping='children' >
    <e-columns>
        <e-column field='FreightID' headerText='Freight ID'
width=130></e-column>
        <e-column field='FreightName' headerText='Freight Name'
width=195></e-column>
        <e-column field='UnitWeight' headerText='Weight Per Unit'
textAlign='Right' type='number' width=130></e-column>
        <e-column field='TotalUnits' headerText='Total Units'
textAlign='Right' type='number' width=125></e-column>
    </e-columns>
    <e-aggregates>
    <e-aggregate [showChildSummary]='true'>
        <e-columns>
            <e-column field="UnitWeight" type="Max">
                <ng-template #footerTemplate let-data>Maximum:
{{data.Max}}</ng-template>
            </e-column>
            <e-column field="TotalUnits" type="Min">
                <ng-template #footerTemplate let-data>Minimum:
{{data.Min}}</ng-template>
            </e-column>
        </e-columns>
    </e-aggregate>
    </e-aggregates>
    </ejs-treegrid>`
  })
export class AppComponent implements OnInit {
  public data?: Object[];
  ngOnInit(): void {
    this.data = summaryRowData;
  }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Footer aggregate in Angular Treegrid component

Footer aggregate value is calculated for all the rows, and it is displayed in the footer cells. Use the [footerTemplate](#) property to render the aggregate value in footer cells.

APP.COMPONENT.TS

```

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

```

```

import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { AggregateService } from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit } from '@angular/core';
import { summaryRowData } from './datasource';
@Component({
  imports: [

    TreeGridModule
  ],
  providers: [AggregateService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' height='240'
[treeColumnIndex]='0' childMapping='children' >
    <e-columns>
        <e-column field='FreightID' headerText='Freight ID'
width=130></e-column>
        <e-column field='FreightName' headerText='Freight Name'
width=195></e-column>
        <e-column field='UnitWeight' headerText='Weight Per Unit'
textAlign='Right' type='number' width=130></e-column>
        <e-column field='TotalUnits' headerText='Total Units'
textAlign='Right' type='number' width=125></e-column>
    </e-columns>
    <e-aggregates>
        <e-aggregate [showChildSummary]='false'>
            <e-columns>
                <e-column field="UnitWeight" type="Max"
columnName='UnitWeight'>
                    <ng-template #footerTemplate let-data>Maximum:
{{data.Max}}</ng-template>
                </e-column>
                <e-column field="TotalUnits" type="Min"
columnName='TotalUnits'>
                    <ng-template #footerTemplate let-data>Minimum:
{{data.Min}}</ng-template>
                </e-column>
            </e-columns>
        </e-aggregate>
    </e-aggregates>
    </ejs-treegrid>`
  })
export class AppComponent implements OnInit {
  public data?: Object[];
  ngOnInit(): void {
    this.data = summaryRowData;
  }
}

```

MAIN.TS

```

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

```

The aggregate values must be accessed inside the template using their corresponding [type](#) name.

Get the footer content element by using [getFooterContent](#) method in the tree grid.

Get the footer content table element by using [getFooterContentTable](#) method in the tree grid.

How to format aggregate value

You can format the aggregate value result by using the [format](#) property.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { AggregateService } from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit } from '@angular/core';
import { summaryData } from './datasource';
@Component({
  imports: [
    TreeGridModule
  ],
  providers: [AggregateService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' height='260' width='auto'
[treeColumnIndex]='0' childMapping='subtasks' >
    <e-columns>
      <e-column field='category' headerText='Category'
width=130></e-column>
      <e-column field='units' headerText='Total Units'
textAlign='Right' type='number' Width=130></e-column>
      <e-column field='unitPrice' headerText='Unit Price($)'
format='C2' textAlign='Right' type='number' width=110 ></e-column>
      <e-column field='price' headerText='Price($)'
textAlign='Right' type='number' width=160 ></e-column>
    </e-columns>
    <e-aggregates >
      <e-aggregate >
        <e-columns>
          <e-column field="price" format='C2' type="Sum"
customAggregate='customAggregateFn'>
            <ng-template #footerTemplate let-data>Total:
{{data.Sum}}</ng-template>
          </e-column>
        </e-columns>
      </e-aggregate>
    </e-aggregates>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  ngOnInit(): void {
    this.data = summaryData;
  }
}
```

MAIN.TS

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

Custom aggregate in Angular Treegrid component

To calculate the aggregate value with your own aggregate functions, use the custom aggregate option.

To use custom aggregation, specify the [type](#) as `Custom`, and provide the custom aggregate function in the [customAggregate](#) property.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { AggregateService } from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit } from '@angular/core';
import { summaryData } from './datasource';
import { getObject, CustomSummaryType } from '@syncfusion/ej2-grids';
@Component({
  imports: [

    TreeGridModule
  ],
  providers: [AggregateService ],
  standalone: true,
  selector: 'app-container',
  template: `

```

```

ngOnInit(): void {
    this.data = summaryData;
}
customAggregateFn (data: Object): number {
    let sampleData: Object[] = getObject('result', data);
    let countLength: number; countLength = 0;
    sampleData.filter((item: Object) => {
        let data: string = getObject('category', item);
        if (data === 'Frozen seafood') {
            countLength++;
        }
    });
    return countLength;
};
}

```

MAIN.TS

```

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

```

To access the custom aggregate value inside the template, use the key as **Custom**.

Tool Bar

Tool bar in Angular Treegrid component

The TreeGrid provides ToolBar support to handle treegrid actions. The [toolbar](#) property accepts either the collection of built-in toolbar items and [ItemModel](#) objects for custom toolbar items or HTML element ID for toolbar template.

To use ToolBar, inject **Toolbar** module in the treegrid.

Enable or disable toolbar items

Enable or disable toolbar items by using the **enableItems** method.

You can also use the [enableToolbarItems](#) method to enable or disable the tool bar items. In this method, you need to pass the toolbar items and isEnabled as parameters.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService, EditService, ToolbarService }
    from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
    imports: [

```

```

        TreeGridModule,
        ButtonModule,
        DropDownListAllModule,
    ],
    providers: [PageService,
                SortService,
                FilterService,
                EditService,
                ToolbarService],
    standalone: true,
    selector: 'app-container',
    template: `
        <button ej-button (click)=>enableClick()>Enable</button>
        <button ej-button (click)=>disableClick()>Disable</button>
        <ejs-treegrid [dataSource]='data' #treegrid height='220'
[allowFiltering]='true' (toolbarClick)=>toolbarClick($event) '
[allowPaging]='true' pageSettings='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
            <e-columns>
                <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
                <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
                <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
            </e-columns>
        </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public toolbarOptions?: ToolbarItems[] | any;
        public pager?: Object;
        @ViewChild('treegrid')
        public treeGridObj?: TreeGridComponent;
        ngOnInit(): void {
            this.data = sampleData;
            this.toolbarOptions = ['QuickFilter', 'ClearFilter'];
            this.pager = { pageSize: 8 }
        }
        toolbarClick(args: Object | any): void{
            if (args.item.text === 'QuickFilter') {
                this.treeGridObj?.filterByColumn('taskName', 'startswith',
'Testing');
            }
            if (args.item.text === 'ClearFilter') {
                this.treeGridObj?.clearFiltering();
            }
        }
        enableClick() {
            this.treeGridObj?.toolbarModule.enableItems([this.treeGridObj?.element.id +
            '_gridcontrol_QuickFilter', this.treeGridObj?.element.id +
            '_gridcontrol_ClearFilter'], true); // enable toolbar items.
        };
        disableClick() {

```

```

this.treeGridObj?.toolbarModule.enableItems([this.treeGridObj?.element.id +
'_gridcontrol_QuickFilter', this.treeGridObj?.element.id +
'_gridcontrol_ClearFilter'], false); // disable toolbar items.
    };
}

```

MAIN.TS

```

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

```

Add toolbar at the bottom of tree grid

Add the toolbar component at the bottom of the tree grid using the [created](#) event.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService, EditService, ToolbarService
} from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    SortService,
    FilterService,
    EditService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' id='TreeGrid' #treegrid
height='220' [treeColumnIndex]='1' childMapping='subtasks'
[toolbar]='toolbarOptions' (created)="created($event)">
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>

```



```

        </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: object[];
        public toolbarOptions?: string[];
        @ViewChild('treegrid')
        public treegrid?: TreeGridComponent;
        ngOnInit(): void {
            this.data = sampleData.slice(0, 2);
            this.toolbarOptions = ['Print', 'Search'];
        }
        created(args: any) {
            let toolbarOptions: HTMLElement = (this.treegrid as
TreeGridComponent).element.querySelector('.e-toolbar') as HTMLElement;
            (this.treegrid as
TreeGridComponent).element.appendChild(toolbarOptions);
        }
    }
}

```

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

- [To know more about Toolbar Component](#)

Refer to Syncfusion [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. Also, explore Syncfusion [Angular Tree Grid example](#) to know how to present and manipulate data.

Tool bar items in Angular Treegrid component

Built-in toolbar items

Built-in toolbar items execute standard actions of the treegrid, and it can be added by defining the [toolbar](#) as a collection of built-in items. It renders the button with icon and text.

The following table shows built-in toolbar items and its actions.

Built-in Toolbar Items	Actions
-----	-----
ExpandAll	Expands all the rows.
CollapseAll	Collapses all the rows.
Add	Adds a new record.
Edit	Edits the selected record.
Update	Updates the edited record.
Delete	Deletes the selected record.

- | Cancel | Cancels the edit state.
- | Search | Searches the records by the given key.
- | Print | Prints the treegrid.
- | ExcelExport | Exports the treegrid to Excel.
- | PdfExport | Exports the treegrid to PDF.
- | WordExport | Exports the treegrid to Word.
- | Indent | Indents the record to one level of hierarchy.
- | Outdent | Outdents the record to one level of hierarchy.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService, EditService, ToolbarService }
  from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    SortService,
    FilterService,
    EditService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' height='220'
[allowPaging]='true' pageSettings='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
```

```

public toolbarOptions?: ToolbarItems[];
public pager?: Object;
ngOnInit(): void {
    this.data = sampleData;
    this.toolbarOptions = ['Print', 'Search'];
    this.pager = { pageSize: 8 }
}
}

```

MAIN.TS

```

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

```

* The [toolbar](#) has options to define both built-in and custom toolbar items.

Custom toolbar component in a specific position

By default, the custom toolbar items are in the left position. Change the position by using the [align](#) property. In the following sample, the right position is applied for the Collapse All toolbar item and left for the Expand All toolbar item.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { ToolbarService } from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit, ViewChild } from '@angular/core';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { sampleData } from './datasource';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridModule
  ],
  providers: [ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data' height='220'
[treeColumnIndex]='1' childMapping='subtasks' [toolbar]='toolbarOptions'
(toolbarClick)='clickHandler($event)'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
  </ejs-treegrid>`
})

```

```

export class AppComponent implements OnInit {
  public data?: object[];
  public toolbarOptions?: object[];
  @ViewChild('treegrid') public treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
    this.toolbarOptions = [{ text: 'Expand All', tooltipText: 'Expand All', prefixIcon: 'e-expand', id: 'expandall' },
      { text: 'Collapse All', tooltipText: 'collection All', prefixIcon: 'e-collapse', id: 'collapseall', align: 'Right' }];
  }
  clickHandler(args: ClickEventArgs): void {
    if (args.item.id === 'expandall') {
      (this.treegrid as TreeGridComponent).expandAll();
    }
    if (args.item.id === 'collapseall') {
      (this.treegrid as TreeGridComponent).collapseAll();
    }
  }
}

```

MAIN.TS

```

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

```

Custom tool bar in Angular Treegrid component

Custom toolbar items can be added by defining the [toolbar](#) as a collection of [ItemModels](#).

Actions for this customized toolbar items are defined in the [toolbarClick](#) event.

By default, Custom toolbar items are in position **Left**. You can change the position by using the [align](#) property. In the below sample, we have applied position **Right** for the **Quick Filter** toolbar item.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService, EditService, ToolbarService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],

```

```

providers: [PageService,
            SortService,
            FilterService,
            EditService,
            ToolbarService],
standalone: true,
selector: 'app-container',
template: `<ejs-treegrid [dataSource]='data' [allowFiltering]='true'
#treegrid height='220' (toolbarClick)='toolbarClick($event)'
[allowPaging]='true' pageSettings='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
</ejs-treegrid>`
}))
export class AppComponent implements OnInit {
    public data?: Object[];
    public toolbarOptions?: ToolbarItems[] | any;
    public pager?: Object;
    @ViewChild('treegrid')
    public treeGridObj?: TreeGridComponent;
    ngOnInit(): void {
        this.data = sampleData;
        this.toolbarOptions = [{text: 'Quick Filter', tooltipText: 'Quick
Filter', id: 'toolbarfilter', align:'Right'}];
        this.pager = { pageSize: 8 }
    }
    toolbarClick(args: Object | any): void {
        if (args.item.id === 'toolbarfilter') {
            (this.treeGridObj as
TreeGridComponent).filterByColumn('taskName', 'startswith', 'Testing');
        }
    }
}

```

MAIN.TS

```

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

```

* The [toolbar](#) has options to define both built-in and custom toolbar items.

* If a toolbar item does not match the built-in items, it will be treated as a custom toolbar item.

Built-in and custom items in toolbar

TreeGrid have an option to use both built-in and custom toolbar items at same time.

In the below example, **ExpandAll**, **CollapseAll** are built-in toolbar items and **Click** is custom toolbar item.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService, EditService, ToolbarService }
  from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    SortService,
    FilterService,
    EditService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
pageSettings='pager' [treeColumnIndex]='1' childMapping='subtasks'
[toolbar]='toolbarOptions'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public toolbarOptions?: ToolbarItems[] | any;
  public pager?: Object;
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
  }
}
```

```

        this.toolbarOptions = ['ExpandAll', 'CollapseAll', { text: 'Click',
tooltipText: 'Click', prefixIcon: 'e-time', id: 'Click' }]];
        this.pager = { pageSize: 8 }
    }
    toolbarClick(args: Object | any): void {
        if (args.item.text === 'Click') {
            alert("Custom toolbar click...");
        }
    }
}

```

MAIN.TS

```

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

```

Edit in Angular TreeGrid component

The TreeGrid component provides powerful options for dynamically inserting, deleting, and updating records, enabling you to modify data directly within the tree grid. This feature is useful when you want to enable you to perform CRUD (Create, Read, Update, Delete) operations seamlessly.

To enable editing functionality directly within the tree grid, you need to configure the [allowEditing](#), [allowAdding](#), and [allowDeleting](#) properties within the [editSettings](#) to **true**.

Editing feature requires a primary key column for CRUD operations. To define the primary key, set [columns.isPrimaryKey](#) to **true** in particular column.

You can start the edit action either by double clicking the particular row or by selecting the required row and click on **Edit** button in the toolbar. Similarly, you can add a new record to tree grid either by clicking on **Add** button in the toolbar or on an external button which is bound to invoke the [addRecord](#) method of the tree grid. **Save** and **Cancel** actions while in edit mode is possible using respective toolbar icon in tree grid. Deletion of the record is possible by selecting the required row and click on **Delete** button in the toolbar.

To use CRUD, inject the [EditService](#) module into the [Link to the Video](#) section.

To learn about what are all the edit modes and edit types are available in Angular TreeGrid, you can check on this video

```
<div style='height:15px'></div>
```

Here's an example of how to enable editing in the tree grid:

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit ,ViewChild} from '@angular/core';
import { sampleData } from './datasource';
import { EditSettingsModel,EditService, ToolbarService, PageService } from
 '@syncfusion/ej2-angular-treegrid';
@Component({
    imports: [

```

```

    TreeGridAllModule,
  ],
  providers: [EditService, ToolbarService, PageService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='270' [editSettings]='editSettings' childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
[isPrimaryKey]='true' textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='priority' headerText='Priority'
textAlign='Right' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
  ))
export class AppComponent implements OnInit {
  public data?: Object[];
  public editSettings?: EditSettingsModel;
  ngOnInit(): void {
    this.data = sampleData;
    this.editSettings = { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Cell' };
  }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date('02/03/2017'),
    endDate: new Date('02/07/2017'),
    progress: 100,
    duration: 5,
    priority: 'Normal',
    approved: false,
    subtasks: [
      { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
      { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
      { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),

```



```

        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ]
},
{
    taskID: 6,
    taskName: 'Design',
    startDate: new Date('02/10/2017'),
    endDate: new Date('02/14/2017'),
    duration: 3,
    progress: 86,
    priority: 'High',
    approved: false,
    subtasks: [
        { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
        { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,

```

```

        progress: 50,
        duration: 11,
        subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
                { taskID: 15, taskName: 'Development Task 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
                progress: '50', priority: 'High', approved: false },
                { taskID: 16, taskName: 'Development Task 2',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
                progress: '50', priority: 'Low', approved: true },
                { taskID: 17, taskName: 'Testing', startDate: new
                Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
                progress: '0', priority: 'Normal', approved: true },
                { taskID: 18, taskName: 'Bug fix', startDate: new
                Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
                progress: '0', priority: 'Critical', approved: false },
                { taskID: 19, taskName: 'Customer review meeting',
                startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
                progress: '0', priority: 'High', approved: false },
                { taskID: 20, taskName: 'Phase 1 complete',
                startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
                progress: '50', priority: 'Low', approved: true }
            ]
        }]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,

```

```

        subtasks: [
            { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
            { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
            { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
            endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
            endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
            { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
            endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
        ]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
                { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),

```

```

        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
  }
}
]
];

```

MAIN.TS

```

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

```

* If [columns.isIdentity](#) is enabled, then it will be considered as a read-only column when adding a record.

* You can disable editing for a particular column, by specifying `columns.allowEditing` to **false**.

* You can use the **Insert** key to add a new row to the tree grid and use the **Delete** key to delete the selected row from the tree grid.

Toolbar with edit option

The toolbar with edit option feature in the TreeGrid component provides a [built-in toolbar](#) that includes various items for executing editing actions. This feature allows you to easily perform edit operations on the tree grid data, such as modifying cell values, updating changes, and canceling edits.

To enable this feature, you need to configure the [toolbar](#) property of the TreeGrid component. This property allows you to define the items that will be displayed in the tree grid toolbar. By including the relevant items like **Edit**, **Add**, **Delete**, **Update**, and **Cancel** within the `toolbar` property, you can enable the edit options in the toolbar.

Here's an example of how to enable the toolbar with edit option in the tree grid:

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';

```

```

import { EditSettingsModel, ToolbarItems, EditService, ToolbarService,
PageService } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  standalone: true,
  providers: [EditService, ToolbarService, PageService],
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
height='270' [toolbar]='toolbar' [editSettings]='editSettings'
childMapping='subtasks' >
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
[isPrimaryKey]='true' textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='priority' headerText='Priority'
textAlign='Right' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
      </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public editSettings?: EditSettingsModel;
  public toolbar?: ToolbarItems[];
  ngOnInit(): void {
    this.data = sampleData;
    this.editSettings = { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Row' };
    this.toolbar = ['Add', 'Edit', 'Delete', 'Update', 'Cancel'];
  }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date('02/03/2017'),
    endDate: new Date('02/07/2017'),
    progress: 100,
    duration: 5,
    priority: 'Normal',
    approved: false,
    subtasks: [
      { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
    ]
  }
]

```

```

        { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
          endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
        { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
          endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
          endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
      ],
    },
    {
      taskID: 6,
      taskName: 'Design',
      startDate: new Date('02/10/2017'),
      endDate: new Date('02/14/2017'),
      duration: 3,
      progress: 86,
      priority: 'High',
      approved: false,
      subtasks: [
        { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
          endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
        { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
          endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
          endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
      ]
    },
    {
      taskID: 12,
      taskName: 'Implementation Phase',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'Normal',
      approved: false,
      duration: 11,
      progress: 66,
      subtasks: [
        {

```

```

        taskID: 13,
        taskName: 'Phase 1',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'High',
        approved: false,
        progress: 50,
        duration: 11,
        subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
                { taskID: 15, taskName: 'Development Task 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'High', approved: false },
                { taskID: 16, taskName: 'Development Task 2',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'Low', approved: true },
                { taskID: 17, taskName: 'Testing', startDate: new
                    Date('02/20/2017'),
                    endDate: new Date('02/21/2017'), duration: 2,
                    progress: '0', priority: 'Normal', approved: true },
                { taskID: 18, taskName: 'Bug fix', startDate: new
                    Date('02/24/2017'),
                    endDate: new Date('02/25/2017'), duration: 2,
                    progress: '0', priority: 'Critical', approved: false },
                { taskID: 19, taskName: 'Customer review meeting',
                    startDate: new Date('02/26/2017'),
                    endDate: new Date('02/27/2017'), duration: 2,
                    progress: '0', priority: 'High', approved: false },
                { taskID: 20, taskName: 'Phase 1 complete',
                    startDate: new Date('02/27/2017'),
                    endDate: new Date('02/27/2017'), duration: 0,
                    progress: '50', priority: 'Low', approved: true }
            ]
        }
    ],
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',

```

```

        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'Critical',
        approved: false,
        duration: 12,
        progress: 90,
        subtasks: [
            { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
            { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
            { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
            endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
            endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
            { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
            endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
        ]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },

```



```

        { taskID: 32, taskName: 'Development Task 2',
        startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
        progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
        Date('02/20/2017'),
          endDate: new Date('02/21/2017'), duration: 2,
        progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
        Date('02/24/2017'),
          endDate: new Date('02/25/2017'), duration: 2,
        progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
        startDate: new Date('02/26/2017'),
          endDate: new Date('02/27/2017'), duration: 2,
        progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
        startDate: new Date('02/27/2017'),
          endDate: new Date('02/27/2017'), duration: 0,
        progress: '50', priority: 'Critical', approved: false },
      ]
    }
  ]
}
];

```

MAIN.TS

```

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

```

Disable editing for particular column

In TreeGrid component, you have an option to disable editing for a specific column. This feature is useful when you want to prevent editing certain columns, such as columns that contain calculated values or read-only data.

To disable editing for a particular column, you can use the [allowEditing](#) property of the **columns** object. By setting this property to **false**, you can prevent editing for that specific column.

Here's an example that demonstrates how to disable editing for the column in the tree grid:

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { Column, EditSettingsModel, ToolbarItems, TreeGridComponent
,EditService, ToolbarService, PageService} from '@syncfusion/ej2-angular-
treegrid';
import { DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns';

```

```

@Component({
  imports: [
    TreeGridAllModule, DropDownListAllModule
  ],
  providers: [EditService, ToolbarService, PageService],
  standalone: true,
  selector: 'app-container',
  template: `<div style="display: flex">
    <label style="padding: 30px 17px 0 0;"> Select column to
disable editing:</label>
    <ejs-dropdownlist #dropdown style="padding: 26px 0 0 0"
index="0" width="150" [dataSource]="dropdownData" [fields]="dropdownFields"
(change)="selectColumn($event)"></ejs-dropdownlist>
  </div>
  <ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' height='230' [toolbar]='toolbar'
[editSettings]='editSettings' childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
[isPrimaryKey]='true' textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task
Name' textAlign='Left' width=180></e-column>
      <e-column field='priority' headerText='Priority'
textAlign='Right' width=90></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public editSettings?: EditSettingsModel;
  public toolbar?: ToolbarItems[];
  @ViewChild('treegrid') public treegrid?: TreeGridComponent;
  @ViewChild('dropdown') public dropdown?: DropDownListComponent;
  public dropdownData: Object[] = [
    { text: 'Task ID', value: 'taskID' },
    { text: 'Task Name', value: 'taskName' },
    { text: 'Priority', value: 'priority' },
    { text: 'Duration', value: 'duration' },
  ];
  public dropdownFields: Object = { text: 'text', value: 'value' }; //
Define fields for the dropdown
  public currentColumn?: any;
  ngOnInit(): void {
    this.data = sampleData;
    this.editSettings = { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Row' };
    this.toolbar = ['Add', 'Edit', 'Delete', 'Update', 'Cancel'];
  }
  selectColumn(args: any) {

    if (this.currentColumn) {
      ((this.treegrid as
TreeGridComponent).columns[this.currentColumn.index] as any).allowEditing =
true;
    }
  }
}

```

```

        // Update the 'allowEditing' property for the selected column
        this.currentColumn = this.treegrid?.getColumnByField((args.value
as string)) as Column;
        this.currentColumn.allowEditing = false;
        this.treegrid?.refreshColumns();
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
                priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
                priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
                priority: 'Normal', approved: false },

```

```

        { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
          endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
          endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
          endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
      ]
    },
    {
      taskID: 12,
      taskName: 'Implementation Phase',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'Normal',
      approved: false,
      duration: 11,
      progress: 66,
      subtasks: [
        {
          taskID: 13,
          taskName: 'Phase 1',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/27/2017'),
          priority: 'High',
          approved: false,
          progress: 50,
          duration: 11,
          subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
              { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
              { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
              { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),

```

```

        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
      taskID: 22,
      taskName: 'Implementation Module 2',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/28/2017'),
      priority: 'Critical',
      approved: false,
      duration: 12,
      progress: 90,
      subtasks: [
        { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
        { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },

```

```

        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
                { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
                { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
            ]
        }
    ]
}
]
];

```

MAIN.TS

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

* If you have set the [isPrimaryKey](#) property to **true** for a column, editing will be automatically disabled for that column.

* You can disable the particular cell using [cellEdit](#) event. Please refer this [link](#).

Disable editing for particular row

In the TreeGrid component, you can prevent editing of specific rows based on certain conditions. This feature is useful when you want to restrict editing for certain rows, such as read-only data, calculated values, or protected information. It helps maintain data integrity and ensures that only authorized changes can be made in the tree grid.

To disable editing for a particular row, use the [actionBegin](#) event of the tree grid based on **requestType** as **beginEdit**. You can then set the **args.cancel** property to **true** to prevent editing for that row.

In the below demo, the rows which are having the value for **Task Name** column as **Testing** or **planning** is prevented from editing.

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems, TreeGridComponent, EditService, ToolbarService,
PageService } from '@syncfusion/ej2-angular-treegrid';
import { EditEventArgs } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [EditService, ToolbarService, PageService],
  standalone: true,
  selector: 'app-container',
  template: ` <ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' height='230' [toolbar]='toolbar'
(actionBegin)="actionBegin($event)" [editSettings]='editSettings'
childMapping='subtasks' >
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
[isPrimaryKey]='true' textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
      <e-column field='priority' headerText='Priority'
editType= 'dropdownedit' width=90></e-column>
    </e-columns>
```

```

        </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {

        public data?: Object[];
        @ViewChild('treegrid') public treegrid?: TreeGridComponent;
        public editSettings: Object = { allowEditing: true, mode: "Row" };
        public toolbar?: ToolbarItems[];
        ngOnInit(): void {
            this.data = sampleData;
            this.toolbar = ['Edit', 'Update', 'Cancel'];
        }

        actionBegin(args: EditEventArgs) {
            if (args.requestType === 'beginEdit' && ((args.rowData as
any).taskName === 'Testing' || (args.rowData as any).taskName === 'Planning'
)) {
                args.cancel = true;
            }
        }
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ],
    },
]

```



```

{
    taskID: 6,
    taskName: 'Design',
    startDate: new Date('02/10/2017'),
    endDate: new Date('02/14/2017'),
    duration: 3,
    progress: 86,
    priority: 'High',
    approved: false,
    subtasks: [
        { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
            endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
        { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
            endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
            endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
            endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
            endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
    ],
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            progress: 50,
            duration: 11,
            subtasks: [{
                taskID: 14,
                taskName: 'Implementation Module 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'Normal',
            }
            ]
        }
    ]
}

```

```

        duration: 11,
        progress: 10,
        approved: false,
        subtasks: [
            { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
            { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
            { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),

```

```

        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  },
  {
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
      taskID: 30,
      taskName: 'Implementation Module 3',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'High',
      approved: false,
      duration: 11,
      progress: 60,
      subtasks: [
        { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
        { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },

```

```

        { taskID: 35, taskName: 'Customer review meeting',
        startDate: new Date('02/26/2017'),
          endDate: new Date('02/27/2017'), duration: 2,
        progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
        startDate: new Date('02/27/2017'),
          endDate: new Date('02/27/2017'), duration: 0,
        progress: '50', priority: 'Critical', approved: false },
      ]
    }
  ]
}
];

```

MAIN.TS

```

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

```

Editing template column

The editing template column feature in the Tree Grid allows you to create custom editing templates for specific columns in the tree grid. This feature is particularly useful when you need to customize the editing experience for certain columns, such as using custom input controls or displaying additional information during editing.

To enable the editing template column feature, you need to define the [field](#) property for the specific column in the tree grid's configuration. The [field](#) property maps the column to the corresponding field name in the data source, allowing you to edit the value of that field.

In the below demo, the **priority** column is rendered with the template.

APP.COMPONENT.TS

```

{% raw %}
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { Column, EditSettingsModel, ToolbarItems, TreeGridComponent } from
 '@syncfusion/ej2-angular-treegrid';
import { DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns';
@Component({
  selector: 'app-container',
  template: `
    <ejs-treegrid #treegrid [dataSource]='data'
    [treeColumnIndex]='1' height='230' [toolbar]='toolbar'
    [editSettings]='editSettings' (actionBegin)="actionBegin($event)"
    childMapping='subtasks' >
    <e-columns>
    <e-column field='taskID' headerText='Task ID' [isPrimaryKey]='true'
    textAlign='Right' width=90></e-column>
    <e-column field='taskName' headerText='Task Name' textAlign='Left'
    width=180></e-column>
    <e-column field='priority' headerText='Priority' width=90>
    <ng-template #template let-data>

```

```

<a href="#">{{data.priority}}</a>
</ng-template>
<ng-template #editTemplate let-data>
<ejs-dropdownlist [dataSource]='selectDatasource'
[(ngModel)]='taskData.priority' [fields]="fields" popupHeight="150px"></ejs-
dropdownlist>
</ng-template>
</e-column>
<e-column field='duration' headerText='Duration' textAlign='Right'
width=80></e-column>
</e-columns>
</ejs-treegrid>`
)})
export class AppComponent implements OnInit {
public data?: Object[];
public editSettings?: EditSettingsModel;
public toolbar?: ToolbarItems[];
@ViewChild('treegrid') public treegrid?: TreeGridComponent;
public selectDatasource = [
{ text: 'High', value: 'High' },
{ text: 'Low', value: 'Low' },
{ text: 'Critical', value: 'Critical' },
{ text: 'Normal', value: 'Normal' },
];
public taskData?: object | any;
public fields = { value: 'value', text: 'text' };
ngOnInit(): void {
this.data = sampleData;
this.editSettings = { allowEditing: true, allowAdding: true, allowDeleting:
true, mode: 'Row' };
this.toolbar = ['Add', 'Edit', 'Delete', 'Update', 'Cancel'];
}
actionBegin(args: any) {
if (args.requestType === 'beginEdit' || args.requestType === 'add') {
this.taskData = Object.assign({}, args.rowData);
}
if (args.requestType === 'save') {
(args.data as any)['priority'] = this.taskData['priority'];
}
}
}
{% enddraw %}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
{
taskID: 1,
taskName: 'Planning',
startDate: new Date('02/03/2017'),
endDate: new Date('02/07/2017'),
progress: 100,
duration: 5,
}
]

```

```

        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
                endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
        ]
    },
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date('02/17/2017'),

```

```

        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,
                    taskName: 'Implementation Module 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'Normal',
                    duration: 11,
                    progress: 10,
                    approved: false,
                    subtasks: [
                        { taskID: 15, taskName: 'Development Task 1',
                            startDate: new Date('02/17/2017'),
                            endDate: new Date('02/19/2017'), duration: 3,
                            progress: '50', priority: 'High', approved: false },
                        { taskID: 16, taskName: 'Development Task 2',
                            startDate: new Date('02/17/2017'),
                            endDate: new Date('02/19/2017'), duration: 3,
                            progress: '50', priority: 'Low', approved: true },
                        { taskID: 17, taskName: 'Testing', startDate: new
                            Date('02/20/2017'),
                            endDate: new Date('02/21/2017'), duration: 2,
                            progress: '0', priority: 'Normal', approved: true },
                        { taskID: 18, taskName: 'Bug fix', startDate: new
                            Date('02/24/2017'),
                            endDate: new Date('02/25/2017'), duration: 2,
                            progress: '0', priority: 'Critical', approved: false },
                        { taskID: 19, taskName: 'Customer review meeting',
                            startDate: new Date('02/26/2017'),
                            endDate: new Date('02/27/2017'), duration: 2,
                            progress: '0', priority: 'High', approved: false },
                        { taskID: 20, taskName: 'Phase 1 complete',
                            startDate: new Date('02/27/2017'),
                            endDate: new Date('02/27/2017'), duration: 0,
                            progress: '50', priority: 'Low', approved: true }
                    ]
                }]
            },
            {
                taskID: 21,
                taskName: 'Phase 2',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/28/2017'),
            }
        ]
    }

```

```

        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
                progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
                progress: '50', priority: 'Critical', approved: true },
                { taskID: 25, taskName: 'Testing', startDate: new
                Date('02/21/2017'),
                endDate: new Date('02/24/2017'), duration: 2,
                progress: '0', priority: 'High', approved: false },
                { taskID: 26, taskName: 'Bug fix', startDate: new
                Date('02/25/2017'),
                endDate: new Date('02/26/2017'), duration: 2,
                progress: '0', priority: 'Low', approved: false },
                { taskID: 27, taskName: 'Customer review meeting',
                startDate: new Date('02/27/2017'),
                endDate: new Date('02/28/2017'), duration: 2,
                progress: '0', priority: 'Critical', approved: true },
                { taskID: 28, taskName: 'Phase 2 complete',
                startDate: new Date('02/28/2017'),
                endDate: new Date('02/28/2017'), duration: 0,
                progress: '50', priority: 'Normal', approved: false }
            ]
        }]
    },
    {
        taskID: 29,
        taskName: 'Phase 3',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,

```



```

        duration: 11,
        progress: 60,
        subtasks: [
            { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
            { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
            { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
        ]
    }
}
];

```

MAIN.TS

```

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

```

Customize delete confirmation dialog

Customizing the delete confirmation dialog in Tree Grid allows you to personalize the appearance, content, and behavior of the dialog that appears when you attempt to delete an item. You can modify properties like header, showCloseIcon, and height to tailor the edit dialog to your specific requirements. Additionally, you can override default localization strings to provide custom text for buttons or other elements within the dialog.

To customize the delete confirmation dialog, you can utilize the [toolbarClick](#) event. This event is triggered when a toolbar item, such as the delete button, is clicked.

* To enable the confirmation dialog for the delete operation in the tree grid, you can set the [showDeleteConfirmDialog](#) property of the `editSettings` configuration to **true**.

* You can refer the tree grid [Default text](#) list for more localization.

The following example that demonstrates how to customize the delete confirmation dialog using the `toolbarClick` event:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { Column, EditSettingsModel, ToolbarItems,
TreeGridComponent, EditService, ToolbarService, PageService } from
'@syncfusion/ej2-angular-treegrid';
import { ClickEventArgs, Item } from '@syncfusion/ej2-angular-navigations';
import { L10n } from '@syncfusion/ej2-base';
L10n.load({
  'en-US': {
    treegrid: {
      'OKButton': 'YES',
      'CancelButton': 'Discard',
      'ConfirmDelete': 'Are you sure you want to delete the selected
Record?'
    }
  }
});
@Component({
  imports: [
    TreeGridAllModule,
  ],
  providers: [EditService, ToolbarService, PageService],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' height='230' [toolbar]='toolbar'
(toolbarClick)="toolbarClick($event)" [editSettings]='editSettings'
childMapping='subtasks' >
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
[isPrimaryKey]='true' textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task
Name' textAlign='Left' width=180></e-column>
        <e-column field='priority' headerText='Priority'
editType= 'dropdownedit' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
      </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public editSettings?: EditSettingsModel;
  public toolbar?: ToolbarItems[];
  @ViewChild('treegrid') public treegrid?: TreeGridComponent;

  ngOnInit(): void {
```

```

        this.data = sampleData;
        this.editSettings = { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Row', showDeleteConfirmDialog: true, };
        this.toolbar = ['Add', 'Edit', 'Delete', 'Update', 'Cancel'];
    }
    toolbarClick(args: ClickEventArgs): void {

        if ((args.item as Item).text === 'Delete') {
            const dialogObj= ((this.treegrid as
TreeGridComponent).editModule as any).dialogObj  ;
            dialogObj.header = 'Delete Confirmation Dialog';
            dialogObj.showCloseIcon = true;
        }
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
                endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
    }
]

```

```

        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
              endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
              endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
              endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
              endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
              endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
        ]
    },
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,
                    taskName: 'Implementation Module 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/27/2017'),
                    priority: 'Normal',
                    duration: 11,
                    progress: 10,
                    approved: false,
                    subtasks: [
                        { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),

```

```

        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
        { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
        { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
      taskID: 22,
      taskName: 'Implementation Module 2',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/28/2017'),
      priority: 'Critical',
      approved: false,
      duration: 12,
      progress: 90,
      subtasks: [
        { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
        { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
          endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },

```

```

        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  }
},
{
  taskID: 29,
  taskName: 'Phase 3',
  startDate: new Date('02/17/2017'),
  endDate: new Date('02/27/2017'),
  priority: 'Normal',
  approved: false,
  duration: 11,
  progress: 30,
  subtasks: [{
    taskID: 30,
    taskName: 'Implementation Module 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'High',
    approved: false,
    duration: 11,
    progress: 60,
    subtasks: [
      { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
      { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
      { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
      { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
      { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
      { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),

```

```

                                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
                                ]
                                }]
                                }
                                ]
                                }
                                ]
];

```

MAIN.TS

```

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

```

Update boolean column value with a single click

The Tree Grid allows you to update a boolean column value with a single click in the normal mode of editing. This feature streamlines the process of toggling boolean values within the tree grid, enhancing interaction and efficiency. This can be achieved through the use of the column template feature.

In the following sample, the `CheckBox` component is rendered as a template in the **approved** column to make it editable with a single click.

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { CheckBoxModule } from '@syncfusion/ej2-angular-buttons';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { Column, EditSettingsModel, ToolbarItems, TreeGridComponent
,EditService, ToolbarService, PageService} from '@syncfusion/ej2-angular-
treegrid';
@Component({
  imports: [
    TreeGridAllModule,CheckBoxModule
  ],
  providers: [EditService, ToolbarService, PageService],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' height='230' [toolbar]='toolbar'
[editSettings]='editSettings' childMapping='subtasks' >
      <e-columns>
        <e-column field='taskID' headerText='Task ID'
[isPrimaryKey]='true' textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task
Name' textAlign='Left' width=180></e-column>
        <e-column field='priority' headerText='Priority'
editType= 'dropdownedit' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>

```

```

        <e-column field="approved" headerText="Approved"
textAlign="Right" width="90">
            <ng-template #template let-data>
                <ejs-checkbox
[ (checked) ]="data.approved"></ejs-checkbox>
            </ng-template>
        </e-column>
    </e-columns>
</ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public editSettings?: EditSettingsModel;
        public toolbar?: ToolbarItems[];
        @ViewChild('treegrid') public treegrid?: TreeGridComponent;

        ngOnInit(): void {
            this.data = sampleData;
            this.editSettings = { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Row', showDeleteConfirmDialog: true, };
            this.toolbar = ['Add', 'Edit', 'Delete', 'Update', 'Cancel'];
        }
    }
}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
        progress: 100,
        duration: 5,
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
                endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),

```



```

        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ],
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
                endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
                endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
                endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
        ]
    },
    {
        taskID: 12,
        taskName: 'Implementation Phase',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        approved: false,
        duration: 11,
        progress: 66,
        subtasks: [
            {
                taskID: 13,
                taskName: 'Phase 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'High',
                approved: false,
                progress: 50,
                duration: 11,
                subtasks: [{
                    taskID: 14,

```

```

        taskName: 'Implementation Module 1',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/27/2017'),
        priority: 'Normal',
        duration: 11,
        progress: 10,
        approved: false,
        subtasks: [
            { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
            { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
            { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),

```

```

        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
        { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  },
  {
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
      taskID: 30,
      taskName: 'Implementation Module 3',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'High',
      approved: false,
      duration: 11,
      progress: 60,
      subtasks: [
        { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
        { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
          endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
          endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },

```

```

        { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
  }
}
];

```

MAIN.TS

```

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

```

Edit enum column

The Tree Grid provides a feature that allows you to edit enum type data in a tree grid column. This is particularly useful when you need to edit enumerated list data efficiently.

In the following example, the `DropDownList` component is rendered within the `editTemplate` for the Project Feedback column using `ngTemplate`. The enumerated list data can be bound to the Project Feedback column using the two-way binding (`@bind-Value`).

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns';
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData } from './datasource';
import { Column, EditSettingsModel, ToolbarItems,
TreeGridComponent, EditService, ToolbarService, PageService } from
 '@syncfusion/ej2-angular-treegrid';
import { SaveEventArgs } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [
    TreeGridAllModule, DropDownListModule
  ],
  providers: [EditService, ToolbarService, PageService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data'
[treeNodeIndex]='1' height='230' [toolbar]='toolbar'

```

```

[editSettings]='editSettings' (actionBegin)="actionBegin($event)"
idMapping='TaskID' parentIdMapping='parentID'>
    <e-columns>
        <e-column field='TaskID' headerText='Task ID'
[isPrimaryKey]='true' textAlign='Right' width=90></e-column>
        <e-column field='TaskName' headerText='Task
Name' textAlign='Left' width=180></e-column>
        <e-column field="Task_FeedbackDetails"
headerText="project Feedback" textAlign="Left" width="120">
            <ng-template #editTemplate let-data>
                <ejs-dropdownlist
[(ngModel)]="taskData.Task_FeedbackDetails" [dataSource]="dropDownEnumValue"
[fields]="dropdownFields" popupHeight="150px"></ejs-dropdownlist>
            </ng-template>
        </e-column>
        <e-column field='Duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`
}))
export class AppComponent implements OnInit {
    public data?: Object[];
    public editSettings?: EditSettingsModel;
    public toolbar?: ToolbarItems[];
    public taskData?: ProjectDetails|any ;
    public dropDownEnumValue: string[] = [];
    @ViewChild('treegrid') public treegrid?: TreeGridComponent;
    public dropdownFields: Object = { text: 'Task_FeedbackDetails', value:
'Task_FeedbackDetails' };
    ngOnInit(): void {
        this.data = projectData;
        this.editSettings = { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Row', showDeleteConfirmDialog: true, };
        this.toolbar = ['Add', 'Edit', 'Delete', 'Update', 'Cancel'];
        this.dropDownEnumValue = Object.keys(Feedback).filter((key: string)
=> !isNaN(Number((Feedback as any)[key])));
    }
    actionBegin(args: SaveEventArgs) {
        if (args.requestType === 'beginEdit' || args.requestType === 'add')
        {
            this.taskData = Object.assign({}, args.rowData);
        }
        if (args.requestType === 'save') {
            (args.data as ProjectDetails)['Task_FeedbackDetails'] =
(this.taskData as ProjectDetails)['Task_FeedbackDetails'];
        }
    }
}
export interface ProjectDetails {
    TaskID: number;
    TaskName: string;
    Task_FeedbackDetails: Feedback;
    Duration: number;
}

export enum Feedback {

```

```

    Positive = 0,
    Negative = 1,
  }

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let projectData: Object[] = [
  { TaskID: 1, TaskName: 'Parent Task 1', StartDate: new
Date('02/23/2023'), Duration: 3, Priority : 'Normal',
  EndDate: new Date('02/27/2023'), Progress: '40',Approved:true ,
Task_FeedbackDetails:'positive',parentID: null},
  { TaskID: 2, TaskName: 'Child Task 1', StartDate: new
Date('02/23/2023'), Duration: 4, Priority : 'Low',
  EndDate: new Date('02/27/2023'), Progress: '40', parentID:
1,Task_FeedbackDetails:'positive', Approved:false},
  { TaskID: 3, TaskName: 'Child Task 2', StartDate: new
Date('02/23/2023'), Duration: 2, Priority : 'Normal',
  EndDate: new Date('02/27/2023'), Progress: '40', parentID: 1,
Task_FeedbackDetails:'negative',Approved:true },
  { TaskID: 4, TaskName: 'Child Task 3', StartDate: new
Date('02/23/2023'), Duration: 2, Priority : 'Low',
  EndDate: new Date('02/27/2023'), Progress:
'40',Task_FeedbackDetails:'negative', parentID: 1 },

  { TaskID: 5, TaskName: 'Parent Task 2', StartDate: new
Date('03/14/2023'), Duration: 6, Priority : 'Normal',
  EndDate: new Date('03/18/2023'), Progress:
'40',Task_FeedbackDetails:'negative',Approved:false , parentID: null},
  { TaskID: 6, TaskName: 'Child Task 1', StartDate: new
Date('03/02/2023'), Duration: 11, Priority : 'High',
  EndDate: new Date('03/06/2023'), Progress: '40',
Task_FeedbackDetails:'negative',parentID: 5,Approved:false },
  { TaskID: 7, TaskName: 'Child Task 2', StartDate: new
Date('03/02/2023'), Duration: 7, Priority : 'Critical',
  EndDate: new Date('03/06/2023'), Progress: '40',
Task_FeedbackDetails:'positive',parentID: 5 },
  { TaskID: 8, TaskName: 'Child Task 3', StartDate: new
Date('03/02/2023'), Duration: 10, Priority : 'Breaker',
  EndDate: new Date('03/06/2023'), Progress: '40',
Task_FeedbackDetails:'negative',parentID: 5,Approved:true },
  { TaskID: 9, TaskName: 'Child Task 4', StartDate: new
Date('03/02/2023'), Duration: 15, Priority : 'High',
  EndDate: new Date('03/06/2023'), Progress: '40',
Task_FeedbackDetails:'positive',parentID: 5 ,Approved:false},

  { TaskID: 10, TaskName: 'Parent Task 3', StartDate: new
Date('03/09/2023'), Duration: 17, Priority : 'Breaker',
  EndDate: new Date('03/13/2023'), Progress:
'40',Task_FeedbackDetails:'positive',Approved:false, parentID: null },
  { TaskID: 11, TaskName: 'Child Task 1', StartDate: new
Date('03/9/2023'), Duration: 0, Priority : 'Low',
  EndDate: new Date('03/13/2023'), Progress:
'40',Task_FeedbackDetails:'negative', parentID: 10 ,Approved:true},

```

```

    { TaskID: 12, TaskName: 'Child Task 2', StartDate: new
Date('03/9/2023'), Duration: 10, Priority : 'Breaker',
    EndDate: new Date('03/13/2023'), Progress:
'40',Task_FeedbackDetails:'negative', parentID: 10 ,Approved:false},
    { TaskID: 13, TaskName: 'Child Task 3', StartDate: new
Date('03/9/2023'), Duration: 11, Priority : 'Normal',
    EndDate: new Date('03/13/2023'), Progress:
'40',Task_FeedbackDetails:'positive', parentID: 10,Approved:false },
    { TaskID: 14, TaskName: 'Child Task 4', StartDate: new
Date('03/9/2023'), Duration: 1, Priority : 'Normal',
    EndDate: new Date('03/13/2023'), Progress:
'40',Task_FeedbackDetails:'positive', parentID: 10,Approved:false },
    { TaskID: 15, TaskName: 'Child Task 5', StartDate: new
Date('03/9/2023'), Duration: 14, Priority : 'Critical',
    EndDate: new Date('03/13/2023'), Progress: '40',
Task_FeedbackDetails:'negative',parentID: 10 ,Approved:true}
];

```

MAIN.TS

```

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

```

Edit complex column

The edit template for complex column in the Tree Grid is used to customize the editing experience when dealing with complex data structures. This capability is particularly useful for handling nested data objects within tree grid columns. By default, the tree grid binds complex data to column fields using the dot (.) operator. However, when you render custom elements, such as input fields, in the edit template for a complex column, you must use the (__) underscore operator instead of the dot (.) operator to bind the complex object.

In the following sample, the input element is rendered in the edit template of the firstName and lastName column. The edited changes can be saved using the `assignee` property of the input element. Since the complex data is bound to the firstName and lastName column, The `assignee` property should be defined as **assignee.firstName and assignee.lastName**, respectively, instead of using the dot notation (`assignee.firstName` and `assignee.lastName`).

APP.COMPONENT.TS

```

import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit ,ViewChild} from '@angular/core';
import { complexData } from './datasource';
import { EditSettingsModel, ToolbarItems ,EditService, ToolbarService,
PageService} from '@syncfusion/ej2-angular-treegrid';
@Component({
    imports: [
        TreeGridAllModule,
    ],
    providers: [EditService, ToolbarService, PageService],
    standalone: true,

```

```

        selector: 'app-container',
        template: `<ejs-treegrid [dataSource]='data' height='250'
[treeColumnIndex]='1' [editSettings]="editSettings" [toolbar]="toolbar"
childMapping='subtasks' >
            <e-columns>
                <e-column field='taskID' [isPrimaryKey]='true'
headerText='Task ID' textAlign='Right' width=90></e-column>
                <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
                <e-column field='assignee.firstName'
headerText='First Name' width=90>
                    <ng-template #editTemplate let-data>
                        <input class="e-input"
name="assignee__firstName" type="text" id="assignee__firstName"
[value]="data.assignee.firstName" />
                    </ng-template>
                </e-column>
                <e-column field='assignee.lastName' headerText='Last
Name' width=90>
                    <ng-template #editTemplate let-data>
                        <input class="e-input"
name="assignee__lastName" type="text" id="assignee__lastName"
[value]="data.assignee.lastName" />
                    </ng-template>
                </e-column>
                <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
            </e-columns>
        </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public editSettings?: EditSettingsModel;
        public toolbar?: ToolbarItems[];
        ngOnInit(): void {
            this.data = complexData;
            this.editSettings = {
                allowEditing: true,
                allowAdding: true,
                allowDeleting: true, mode: 'Row'
            };
            this.toolbar = ['Add', 'Edit', 'Delete', 'Update', 'Cancel'];
        }
    }

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let complexData: Object[] = [
    {
        taskID: 1,
        taskName: 'Planning',
        startDate: new Date('02/03/2017'),
        endDate: new Date('02/07/2017'),
    }
]

```



```

        progress: 100,
        duration: 5,
        assignee: {firstName: 'Nancy', lastName: 'Davolio'},
        priority: 'Normal',
        approved: false,
        subtasks: [
            { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'), assignee: {firstName: 'Andrew', lastName: 'Fuller'},
            endDate: new Date('02/07/2017'), duration: 5, progress: 100,
            priority: 'Normal', approved: false },
            { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'), assignee: {firstName: 'Janet', lastName: 'Leverling'},
            endDate: new Date('02/07/2017'), duration: 5, progress: 100,
            priority: 'Low', approved: true },
            { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'), assignee: {firstName: 'Margaret', lastName: 'Peacock'},
            endDate: new Date('02/07/2017'), duration: 5, progress: 100,
            priority: 'Critical', approved: false },
            { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'), assignee: {firstName: 'Steven', lastName: 'Buchanan'},
            endDate: new Date('02/07/2017'), duration: 0, progress: 0,
            priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 6,
        taskName: 'Design',
        startDate: new Date('02/10/2017'),
        endDate: new Date('02/14/2017'),
        duration: 3,
        progress: 86,
        assignee: {firstName: 'Michael', lastName: 'Suyama'},
        priority: 'High',
        approved: false,
        subtasks: [
            { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'), assignee: {firstName: 'Robert', lastName: 'King'},
            endDate: new Date('02/12/2017'), duration: 3, progress: 60,
            priority: 'Normal', approved: false },
            { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'), assignee: {firstName: 'Laura', lastName: 'Challahan'},
            endDate: new Date('02/12/2017'), duration: 3, progress: 100,
            priority: 'Critical', approved: false },
            { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'), assignee: {firstName: 'Anne', lastName: 'Dodsworth'},
            endDate: new Date('02/14/2017'), duration: 2, progress: 100,
            priority: 'Low', approved: true },
            { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'), assignee: {firstName: 'Tamer', lastName: 'Nancy'},
            endDate: new Date('02/14/2017'), duration: 2, progress: 100,
            priority: 'High', approved: true },
            { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'), assignee: {firstName: 'Laura', lastName: 'Martin'},
            endDate: new Date('02/14/2017'), duration: 0, progress: 0,
            priority: 'Normal', approved: true }
        ]
    }
}

```

```
];
```

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 perform CRUD action externally

Performing CRUD (Create, Read, Update, Delete) actions externally in the Tree Grid allows you to manipulate tree grid data outside the tree grid itself. This can be useful in scenarios where you want to manage data operations programmatically.

Using separate toolbar

The Tree Grid enables external CRUD operations, allowing you to efficiently manage data manipulation within the tree grid. This capability is particularly useful when you need to manage data operations using a separate toolbar.

To perform CRUD operations externally, the following methods are available:

[addRecord](#) - To add a new record. If no data is passed then add form will be shown.

[startEdit](#) - To edit the selected row.

[deleteRecord](#) - To delete a selected row.

[endEdit](#) - If the tree grid is in editable state, then you can save a record by invoking this method.

[closeEdit](#) - To cancel the edited state.

The following example demonstrates the integration of the tree grid with a separate toolbar for external CRUD operations. The toolbar contains buttons for Add, Edit, Delete, Update, and Cancel.

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { Column, EditSettingsModel, ToolbarItems,
TreeGridComponent, EditService, ToolbarService, PageService } from
'@syncfusion/ej2-angular-treegrid';
import { ClickEventArgs } from '@syncfusion/ej2-angular-navigations';
import { ToolbarModule } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    TreeGridAllModule, ToolbarModule
  ],
  standalone: true,
  providers: [EditService, PageService],
  selector: 'app-container',
  template: ` <div style="display: flex">
      <ejs-toolbar (clicked)="onToolbarClick($event)">
        <e-items>
          <e-item text="Add" id="add"></e-item>
```

```

        <e-item text="Edit" id="edit"></e-item>
        <e-item text="Delete" id="delete"></e-item>
        <e-item text="Update" id="update"></e-item>
        <e-item text="Cancel" id="cancel"></e-item>
    </e-items>
</ejs-toolbar>
</div>
<ejs-treegrid #treegrid [dataSource]='data'
[treeColumnIndex]='1' height='230' [editSettings]='editSettings'
childMapping='subtasks' >
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
[isPrimaryKey]='true' textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='priority' headerText='Priority'
editType= 'dropdownedit' width=90></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=80></e-column>
    </e-columns>
</ejs-treegrid>`
}))
export class AppComponent implements OnInit {
    public data?: Object[];
    public editSettings?: EditSettingsModel;
    @ViewChild('treegrid') public treegrid?: TreeGridComponent;

    ngOnInit(): void {
        this.data = sampleData;
        this.editSettings = { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Row', showDeleteConfirmDialog: true, };
    }
    public onToolbarClick(args: ClickEventArgs): void {
        switch ((args as any).item.id) {
            case 'add':
                (this.treegrid as TreeGridComponent).addRecord();
                break;
            case 'edit':
                (this.treegrid as TreeGridComponent).startEdit();
                break;
            case 'delete':
                (this.treegrid as TreeGridComponent).deleteRecord();
                break;
            case 'update':
                (this.treegrid as TreeGridComponent).endEdit();
                break;
            case 'cancel':
                (this.treegrid as TreeGridComponent).closeEdit();
                break;
        }
    }
}

```

DATASOURCE.TS

```
/**
```

```
/* TreeGrid DataSource */
export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date('02/03/2017'),
    endDate: new Date('02/07/2017'),
    progress: 100,
    duration: 5,
    priority: 'Normal',
    approved: false,
    subtasks: [
      { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
      { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
      { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
      { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 6,
    taskName: 'Design',
    startDate: new Date('02/10/2017'),
    endDate: new Date('02/14/2017'),
    duration: 3,
    progress: 86,
    priority: 'High',
    approved: false,
    subtasks: [
      { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
      { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
      { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
      { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
    ]
  }
]
```

```

        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
          endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
      ]
    },
    {
      taskID: 12,
      taskName: 'Implementation Phase',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'Normal',
      approved: false,
      duration: 11,
      progress: 66,
      subtasks: [
        {
          taskID: 13,
          taskName: 'Phase 1',
          startDate: new Date('02/17/2017'),
          endDate: new Date('02/27/2017'),
          priority: 'High',
          approved: false,
          progress: 50,
          duration: 11,
          subtasks: [{
            taskID: 14,
            taskName: 'Implementation Module 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'Normal',
            duration: 11,
            progress: 10,
            approved: false,
            subtasks: [
              { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
              { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
              { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
              { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
              { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
              { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),

```

```

        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
  }],
},
{
  taskID: 21,
  taskName: 'Phase 2',
  startDate: new Date('02/17/2017'),
  endDate: new Date('02/28/2017'),
  priority: 'High',
  approved: false,
  duration: 12,
  progress: 60,
  subtasks: [{
    taskID: 22,
    taskName: 'Implementation Module 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'Critical',
    approved: false,
    duration: 12,
    progress: 90,
    subtasks: [
      { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
      { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
      { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
      { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
      { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
      { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  }]
},
{
  taskID: 29,
  taskName: 'Phase 3',
  startDate: new Date('02/17/2017'),
  endDate: new Date('02/27/2017'),
  priority: 'Normal',

```

```

        approved: false,
        duration: 11,
        progress: 30,
        subtasks: [{
            taskID: 30,
            taskName: 'Implementation Module 3',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            duration: 11,
            progress: 60,
            subtasks: [
                { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
                { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
                { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
                endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
                { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
                endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
                { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
                endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
                { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
                endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
            ]
        }]
    }
]
}
];

```

MAIN.TS

```

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

```

Using external form

Performing the edit operation in a custom external form in the Tree Grid is a valuable feature when you need to customize the edit operation within a separate form rather than the default in tree grid editing.

To enable the use of an external form for editing in tree grid, you can make use of the **RowSelected** property. This property specifies whether the edit operation should be triggered when a row is selected.

In the following example, it demonstrates how to edit the form using an external form by utilizing the **RowSelected** property:

APP.COMPONENT.TS

```
import { NgModule, } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeGridAllModule } from '@syncfusion/ej2-angular-treegrid';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { Column, EditSettingsModel, ToolbarItems,
TreeGridComponent, EditService, ToolbarService, PageService } from
 '@syncfusion/ej2-angular-treegrid';
import { RowSelectEventArgs } from '@syncfusion/ej2-angular-grids';
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns';
import { TextBoxModule, NumericTextBoxAllModule } from '@syncfusion/ej2-
angular-inputs';
import { ButtonModule } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [
    TreeGridAllModule, ButtonModule,
    NumericTextBoxAllModule, TextBoxModule,
    DropDownListModule
  ],
  providers: [EditService, ToolbarService, PageService],
  standalone: true,
  selector: 'app-container',
  template: ` <div class="row" >
    <div class="col-xs-6 col-md-3">
      <div>
        <div class="form-row">
          <div class="form-group col-md-12">
            <label for="taskedit">Task ID</label>
            <input class="form-control"
[ (ngModel) ]="selectedProduct.taskID" type="number" disabled />
          </div>
        </div>
        <div class="form-row">
          <div class="form-group col-md-12">
            <label for="tasknameedit">Task Name</label>
            <ejs-textbox
[ (value) ]="selectedProduct.taskName"></ejs-textbox>
          </div>
        </div>
        <div class="form-row">
          <div class="form-group col-md-12">
            <label for="durationedit">Duration</label>
            <ejs-numerictextbox
[ (value) ]="selectedProduct.duration"></ejs-numerictextbox>
          </div>
        </div>
        <div class="form-row">
          <div class="form-group col-md-12">
            <label for="priorityedit">Priority</label>
```



```

<ejs-dropdownlist
  [(value)]="selectedProduct.priority" [dataSource]="dropdown"
  [fields]="dropdownFields"></ejs-dropdownlist>
</div>
</div>
</div>
<button ej-button id="btn"
(click)="save()">Save</button>
</div>

```

```
taskName?: string;  
duration?: number;  
priority?: string;  
}
```

DATASOURCE.TS

```
/**  
 * TreeGrid DataSource  
 */  
export let sampleData: Object[] = [  
  {  
    taskID: 1,  
    taskName: 'Planning',  
    startDate: new Date('02/03/2017'),  
    endDate: new Date('02/07/2017'),  
    progress: 100,  
    duration: 5,  
    priority: 'Normal',  
    approved: false,  
    subtasks: [  
      { taskID: 2, taskName: 'Plan timeline', startDate: new  
Date('02/03/2017'),  
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,  
priority: 'Normal', approved: false },  
      { taskID: 3, taskName: 'Plan budget', startDate: new  
Date('02/03/2017'),  
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,  
priority: 'Low', approved: true },  
      { taskID: 4, taskName: 'Allocate resources', startDate: new  
Date('02/03/2017'),  
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,  
priority: 'Critical', approved: false },  
      { taskID: 5, taskName: 'Planning complete', startDate: new  
Date('02/07/2017'),  
        endDate: new Date('02/07/2017'), duration: 0, progress: 0,  
priority: 'Low', approved: true }  
    ],  
  },  
  {  
    taskID: 6,  
    taskName: 'Design',  
    startDate: new Date('02/10/2017'),  
    endDate: new Date('02/14/2017'),  
    duration: 3,  
    progress: 86,  
    priority: 'High',  
    approved: false,  
    subtasks: [  
      { taskID: 7, taskName: 'Software Specification', startDate: new  
Date('02/10/2017'),  
        endDate: new Date('02/12/2017'), duration: 3, progress: 60,  
priority: 'Normal', approved: false },  
      { taskID: 8, taskName: 'Develop prototype', startDate: new  
Date('02/10/2017'),
```

```

        endDate: new Date('02/12/2017'), duration: 3, progress: 100,
        priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
        new Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
        Date('02/13/2017'),
        endDate: new Date('02/14/2017'), duration: 2, progress: 100,
        priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
        Date('02/14/2017'),
        endDate: new Date('02/14/2017'), duration: 0, progress: 0,
        priority: 'Normal', approved: true }
    ]
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            progress: 50,
            duration: 11,
            subtasks: [{
                taskID: 14,
                taskName: 'Implementation Module 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'Normal',
                duration: 11,
                progress: 10,
                approved: false,
                subtasks: [
                    { taskID: 15, taskName: 'Development Task 1',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'High', approved: false },
                    { taskID: 16, taskName: 'Development Task 2',
                    startDate: new Date('02/17/2017'),
                    endDate: new Date('02/19/2017'), duration: 3,
                    progress: '50', priority: 'Low', approved: true },
                    { taskID: 17, taskName: 'Testing', startDate: new
                    Date('02/20/2017'),
                    endDate: new Date('02/21/2017'), duration: 2,
                    progress: '0', priority: 'Normal', approved: true },
                ]
            }
        ]
    }
]
}

```

```

        { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
        { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
    ]
  },
  {
    taskID: 21,
    taskName: 'Phase 2',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/28/2017'),
    priority: 'High',
    approved: false,
    duration: 12,
    progress: 60,
    subtasks: [{
      taskID: 22,
      taskName: 'Implementation Module 2',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/28/2017'),
      priority: 'Critical',
      approved: false,
      duration: 12,
      progress: 90,
      subtasks: [
        { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
        { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),

```

```

        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  }],
},
{
  taskID: 29,
  taskName: 'Phase 3',
  startDate: new Date('02/17/2017'),
  endDate: new Date('02/27/2017'),
  priority: 'Normal',
  approved: false,
  duration: 11,
  progress: 30,
  subtasks: [{
    taskID: 30,
    taskName: 'Implementation Module 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'High',
    approved: false,
    duration: 11,
    progress: 60,
    subtasks: [
      { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
      { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
      { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
      { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
      { taskID: 35, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
        endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
      { taskID: 36, taskName: 'Phase 3 complete',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Critical', approved: false },
    ]
  }]
}
]
}
];

```

MAIN.TS

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

Troubleshoot editing works only for first row

The Editing functionalities can be performed based upon the primary key value of the selected row. If [isPrimaryKey](#) property is not defined in the tree grid, then edit or delete action take places the first row. To overcome this, ensure that you establish the [isPrimaryKey](#) property as **true** for the relevant column responsible for holding the unique identifier for each row.

How to make a tree grid column always editable

To make a tree grid column always editable, you can utilize the column template feature of the tree grid. This feature is useful when you want to edit a particular column's values directly within the tree grid.

In the following example, the textbox is rendered in the **Duration** column using a column template. The keyup event for the tree grid is bound using the [created](#) event of the tree grid, and the edited changes are saved in the data source using the [updateRow](#) method of the tree grid.

APP.COMPONENT.TS

```
{% raw %}
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { Column, EditSettingsModel, ToolbarItems, TreeGridComponent } from
 '@syncfusion/ej2-angular-treegrid';
import { RowSelectEventArgs, parentsUntil } from '@syncfusion/ej2-angular-
 grids';
@Component({
  selector: 'app-container',
  template: ` <ejs-treegrid #treegrid [dataSource]='data'
 [treeColumnIndex]='1' height='230' (created)="created()"
 [editSettings]='editSettings' childMapping='subtasks' >
 <e-columns>
 <e-column field='taskId' headerText='Task ID' [isPrimaryKey]='true'
 textAlign='Right' width=90></e-column>
 <e-column field='taskName' headerText='Task Name' textAlign='Left'
 width=180></e-column>
 <e-column field='duration' headerText='Duration' textAlign='Right' width=80>
 <ng-template #template let-data>
 <input id='{{data.taskID}}' value='{{data.duration}}' class='custemp'
 type='text' style='width: 100%'>
 </ng-template>
 </e-column>
 <e-column field='priority' headerText='Priority' editType= 'dropdownedit'
 width=90></e-column>
 </e-columns>
 </ejs-treegrid> `
})
export class AppComponent implements OnInit {
  public data?: Object[];
  @ViewChild('treegrid') public treegrid?: TreeGridComponent;
  public editSettings: Object = { allowEditing: true, mode: "Row" };
  ngOnInit(): void {
    this.data = sampleData;
```

```

}
created() {
  (this.treegrid as TreeGridComponent).element.addEventListener('keyup',
  function (e) { // Bind the keyup event for the grid.
    if ((e.target as HTMLElement).classList.contains('custemp')) { // Based on
    this condition, you can find whether the target is an input element or not.
    var row = parentsUntil(e.target as HTMLElement, 'e-row');
    var rowIndex = (row as HTMLFormElement)['rowIndex']; // Get the row index.
    var uid = row.getAttribute('data-uid');
    var treegrid = (document.getElementsByClassName('e-treegrid')[0] as
    HTMLFormElement)['ej2_instances'][0];
    var rowData = treegrid.getRowObjectFromUID(uid).data; // Get the row data.
    rowData.Freight = ((e.target as HTMLFormElement)['value']); // Update the
    new value for the corresponding column.
    treegrid.updateRow(rowIndex, rowData); // Update the modified value in the
    row data.
  }
  });
}
}
}
{% enddraw %}

```

DATASOURCE.TS

```

/**
 * TreeGrid DataSource
 */
export let sampleData: Object[] = [
  {
    taskID: 1,
    taskName: 'Planning',
    startDate: new Date('02/03/2017'),
    endDate: new Date('02/07/2017'),
    progress: 100,
    duration: 5,
    priority: 'Normal',
    approved: false,
    subtasks: [
      { taskID: 2, taskName: 'Plan timeline', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Normal', approved: false },
      { taskID: 3, taskName: 'Plan budget', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Low', approved: true },
      { taskID: 4, taskName: 'Allocate resources', startDate: new
Date('02/03/2017'),
        endDate: new Date('02/07/2017'), duration: 5, progress: 100,
priority: 'Critical', approved: false },
      { taskID: 5, taskName: 'Planning complete', startDate: new
Date('02/07/2017'),
        endDate: new Date('02/07/2017'), duration: 0, progress: 0,
priority: 'Low', approved: true }
    ],
  },
]

```

```

{
    taskID: 6,
    taskName: 'Design',
    startDate: new Date('02/10/2017'),
    endDate: new Date('02/14/2017'),
    duration: 3,
    progress: 86,
    priority: 'High',
    approved: false,
    subtasks: [
        { taskID: 7, taskName: 'Software Specification', startDate: new
Date('02/10/2017'),
            endDate: new Date('02/12/2017'), duration: 3, progress: 60,
priority: 'Normal', approved: false },
        { taskID: 8, taskName: 'Develop prototype', startDate: new
Date('02/10/2017'),
            endDate: new Date('02/12/2017'), duration: 3, progress: 100,
priority: 'Critical', approved: false },
        { taskID: 9, taskName: 'Get approval from customer', startDate:
new Date('02/13/2017'),
            endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'Low', approved: true },
        { taskID: 10, taskName: 'Design Documentation', startDate: new
Date('02/13/2017'),
            endDate: new Date('02/14/2017'), duration: 2, progress: 100,
priority: 'High', approved: true },
        { taskID: 11, taskName: 'Design complete', startDate: new
Date('02/14/2017'),
            endDate: new Date('02/14/2017'), duration: 0, progress: 0,
priority: 'Normal', approved: true }
    ],
},
{
    taskID: 12,
    taskName: 'Implementation Phase',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 66,
    subtasks: [
        {
            taskID: 13,
            taskName: 'Phase 1',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/27/2017'),
            priority: 'High',
            approved: false,
            progress: 50,
            duration: 11,
            subtasks: [{
                taskID: 14,
                taskName: 'Implementation Module 1',
                startDate: new Date('02/17/2017'),
                endDate: new Date('02/27/2017'),
                priority: 'Normal',
            }
            ]
        }
    ]
}

```



```

        duration: 11,
        progress: 10,
        approved: false,
        subtasks: [
            { taskID: 15, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'High', approved: false },
            { taskID: 16, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
            endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
            { taskID: 17, taskName: 'Testing', startDate: new
Date('02/20/2017'),
            endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Normal', approved: true },
            { taskID: 18, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
            endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: false },
            { taskID: 19, taskName: 'Customer review meeting',
startDate: new Date('02/26/2017'),
            endDate: new Date('02/27/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
            { taskID: 20, taskName: 'Phase 1 complete',
startDate: new Date('02/27/2017'),
            endDate: new Date('02/27/2017'), duration: 0,
progress: '50', priority: 'Low', approved: true }
        ]
    },
    {
        taskID: 21,
        taskName: 'Phase 2',
        startDate: new Date('02/17/2017'),
        endDate: new Date('02/28/2017'),
        priority: 'High',
        approved: false,
        duration: 12,
        progress: 60,
        subtasks: [{
            taskID: 22,
            taskName: 'Implementation Module 2',
            startDate: new Date('02/17/2017'),
            endDate: new Date('02/28/2017'),
            priority: 'Critical',
            approved: false,
            duration: 12,
            progress: 90,
            subtasks: [
                { taskID: 23, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
                endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Normal', approved: true },
                { taskID: 24, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),

```

```

        endDate: new Date('02/20/2017'), duration: 4,
progress: '50', priority: 'Critical', approved: true },
        { taskID: 25, taskName: 'Testing', startDate: new
Date('02/21/2017'),
        endDate: new Date('02/24/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },
        { taskID: 26, taskName: 'Bug fix', startDate: new
Date('02/25/2017'),
        endDate: new Date('02/26/2017'), duration: 2,
progress: '0', priority: 'Low', approved: false },
        { taskID: 27, taskName: 'Customer review meeting',
startDate: new Date('02/27/2017'),
        endDate: new Date('02/28/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 28, taskName: 'Phase 2 complete',
startDate: new Date('02/28/2017'),
        endDate: new Date('02/28/2017'), duration: 0,
progress: '50', priority: 'Normal', approved: false }
    ]
  },
  {
    taskID: 29,
    taskName: 'Phase 3',
    startDate: new Date('02/17/2017'),
    endDate: new Date('02/27/2017'),
    priority: 'Normal',
    approved: false,
    duration: 11,
    progress: 30,
    subtasks: [{
      taskID: 30,
      taskName: 'Implementation Module 3',
      startDate: new Date('02/17/2017'),
      endDate: new Date('02/27/2017'),
      priority: 'High',
      approved: false,
      duration: 11,
      progress: 60,
      subtasks: [
        { taskID: 31, taskName: 'Development Task 1',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Low', approved: true },
        { taskID: 32, taskName: 'Development Task 2',
startDate: new Date('02/17/2017'),
        endDate: new Date('02/19/2017'), duration: 3,
progress: '50', priority: 'Normal', approved: false },
        { taskID: 33, taskName: 'Testing', startDate: new
Date('02/20/2017'),
        endDate: new Date('02/21/2017'), duration: 2,
progress: '0', priority: 'Critical', approved: true },
        { taskID: 34, taskName: 'Bug fix', startDate: new
Date('02/24/2017'),
        endDate: new Date('02/25/2017'), duration: 2,
progress: '0', priority: 'High', approved: false },

```

```

        { taskID: 35, taskName: 'Customer review meeting',
        startDate: new Date('02/26/2017'),
          endDate: new Date('02/27/2017'), duration: 2,
        progress: '0', priority: 'Normal', approved: true },
        { taskID: 36, taskName: 'Phase 3 complete',
        startDate: new Date('02/27/2017'),
          endDate: new Date('02/27/2017'), duration: 0,
        progress: '50', priority: 'Critical', approved: false },
      ]
    }
  ]
}
];

```

MAIN.TS

```

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

```

* If a template column has a corresponding `field` property defined, the value entered in the template column's input field will be stored in the associated edit column of the row's data object.

See also

- [Cascading DropDownList with Tree Grid Editing](#)

Print in Angular Treegrid component

To print the TreeGrid, use the [print](#) method from treegrid instance. The print option can be displayed on the [toolbar](#) by adding the `print` toolbar item.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService, EditService, ToolbarService }
  from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    SortService,

```

```

        FilterService,
        EditService,
        ToolbarService],
standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' height='265'
[treeColumnIndex]='1' childMapping='subtasks' [toolbar]='toolbarOptions'>
  <e-columns>
    <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
    <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
    <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
    <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
  </e-columns>
</ejs-treegrid>`
}))
export class AppComponent implements OnInit {
  public data?: Object[];
  public toolbarOptions?: ToolbarItems[];
  ngOnInit(): void {
    this.data = sampleData;
    this.toolbarOptions = ['Print'];
  }
}

```

MAIN.TS

```

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

```

Page setup

Some of the print options cannot be configured through JavaScript code. So, you have to customize the layout, paper size, and margin options using the browser page setup dialog. Please refer to the following links to know more about the browser page setup:

- [Chrome](#)
- [Firefox](#)
- [Safari](#)
- [IE](#)

Print using an external button

To print the treegrid from an external button, invoke the [print](#) method.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'

```

```

import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `
    <button ej2-button cssClass="btn btn-default" content="Print"
    (click)="onClicked()"></button>
    <ejs-treegrid [dataSource]='data' height='265' #treegrid
    [treeColumnIndex]='1' childMapping='subtasks'>
      <e-columns>
        <e-column field='taskID' headerText='Task ID' textAlign='Right'
        width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
        textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
        textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
        textAlign='Right' width=110></e-column>
      </e-columns>
    </ejs-treegrid>`
  })
export class AppComponent implements OnInit {
  public data?: Object[];
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
  }
  onClicked(): void {
    (this.treeGridObj as TreeGridComponent).print();
  }
}

```

MAIN.TS

```

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

```

Print the visible page

By default, the treegrid prints all the pages. To print the current page alone, set the [printMode](#) to [CurrentPage](#).

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService, EditService, ToolbarService }
  from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit } from '@angular/core';
import { sampleData } from '../datasource';
import { ToolbarItems } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    SortService,
    FilterService,
    EditService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' height='220'
[allowPaging]='true' pageSettings='pager' printMode='CurrentPage'
[treeColumnIndex]='1' childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public toolbarOptions?: ToolbarItems[];
  public pager?: Object;
  ngOnInit(): void {
    this.data = sampleData;
    this.toolbarOptions = ['Print'];
    this.pager = { pageSize: 8 }
  }
}
```

MAIN.TS

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

Print large number of columns

By default, the browser uses A4 as page size option to print pages and to adapt the size of the page the browser print preview will auto-hide the overflowed contents. Hence treegrid with large number of columns will cut off to adapt the print page.

To show large number of columns when printing, adjust the scale option from print option panel based on your content size.

```
<!-- markdownlint-disable MD033 -->
```

```

```

```
<!-- markdownlint-enable MD033 -->
```

Show or Hide columns while Printing

You can show a hidden column or hide a visible column while printing the treegrid using [toolbarClick](#) and [printComplete](#) events.

In the `toolbarClick` event, based on `args.item.text` as `Print`. We can show or hide columns by setting `column.visible` property to `true` or `false` respectively.

In the `printComplete` event, We have reversed the state back to the previous state.

In the below example, we have `Duration` as a hidden column in the treegrid. While printing, we have changed `Duration` to visible column and `StartDate` as hidden column.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService, EditService, ToolbarService }
  from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { Column, ToolbarItems, TreeGridComponent } from '@syncfusion/ej2-
angular-treegrid';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    SortService,
    FilterService,
```

```

        EditService,
        ToolbarService],
standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid [dataSource]='data'
(toolbarClick)='toolbarClick($event)' #treegrid height='265'
[allowPaging]='true' pageSettings='pager' printMode='CurrentPage'
[treeColumnIndex]='1'
    (printComplete)='printComplete()' childMapping='subtasks'
[toolbar]='toolbarOptions'>
        <e-columns>
            <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
            <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
            <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
            <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
        </e-columns>
    </ejs-treegrid>`
))
export class AppComponent implements OnInit {
    public data?: Object[];
    public toolbarOptions?: ToolbarItems[];
    @ViewChild('treegrid')
    public treeGridObj?: TreeGridComponent;
    ngOnInit(): void {
        this.data = sampleData;
        this.toolbarOptions = ['Print'];
    }
    toolbarClick(args: ClickEventArgs): void {
        if (args.item.text === 'Print') {
            let cols: Column[] = (this.treeGridObj as
TreeGridComponent).columns as Column[];
            for (var i = 0; i < cols.length; i++) {
                if (cols[i].field == "duration") {
                    cols[i].visible = true;
                }
                else if (cols[i].field == "startDate") {
                    cols[i].visible = false;
                }
            }
        }
    }
    printComplete(): void {
        let cols: Column[] = (this.treeGridObj as TreeGridComponent).columns
as Column[];
        for (var i = 0; i < cols.length; i++) {
            if (cols[i].field == "duration") {
                cols[i].visible = false;
            }
            else if (cols[i].field == "startDate") {
                cols[i].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));
```

Limitations of Printing Large Data

When treegrid contains large number of data, printing all the data at once is not a best option for the browser performance. Because to render all the DOM elements in one page will produce performance issues in the browser. It leads to browser slow down or browser hang.

If printing of all the data is still needed, we suggest to Export the treegrid to **Excel** or **CSV** or **Pdf** and then print it from another non-web based application.

You can refer to our [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Adaptive in Angular Treegrid component

The Tree Grid user interface (UI) was redesigned to provide an optimal viewing experience and improve usability on small screens.

Render adaptive dialogs

When you enable the [enableAdaptiveUI](#) property, the tree grid will render the filter, sort, and edit dialogs in full screen for a better user experience. The following demo demonstrates this behaviour.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService, EditService,
ToolbarService, AggregateService } from '@syncfusion/ej2-angular-treegrid'
import { Component, OnInit, ViewChild } from '@angular/core';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { sampleData } from './datasource';
@Component({
  imports: [
    TreeGridModule
  ],
  providers: [PageService,
    SortService,
    FilterService,
    EditService,
    ToolbarService,
    AggregateService],
  standalone: true,
  selector: 'app-container',
  template: `<div class="e-adaptive-demo e-bigger">
    <div class="e-mobile-layout">
      <div class="e-mobile-content">
```

```

        <ejs-treegrid #adaptive id="adaptivebrowser"
[dataSource]='data' childMapping='subtasks' [treeColumnIndex]='1'
enableAdaptiveUI='true'
        height='100%' allowPaging='true' allowFiltering='true'
        allowSorting='true' [editSettings]='editSettings'
        [filterSettings]='filterSettings' [toolbar]='toolbar'
(load)='onLoad($event)'>
        <e-columns>
            <e-column field='taskID' headerText='Task ID'
isPrimaryKey='true' width='135' textAlign='Right'></e-column>
            <e-column field='taskName' headerText='Task Name'
width='280'></e-column>
            <e-column field='duration' headerText='Duration'
width='140' textAlign='Right'></e-column>
            <e-column field='progress' headerText='Progress'
width='145' textAlign='Right'></e-column>
        </e-columns>
        </ejs-treegrid>
    </div>
</div>`
    })
    export class AppComponent implements OnInit {
        @ViewChild('adaptive')
        public treegrid?: TreeGridComponent;
        public data?: object[];
        public editSettings?: Object;
        public toolbar?: string[];
        public orderidrules?: Object;
        public customeridrules?: Object;
        public filterSettings?: Object;
        public menuFilter?: Object;
        public checkboxFilter?: Object;
        ngOnInit(): void {
            this.data = sampleData;
            this.editSettings = { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Dialog' };
            this.toolbar = ['Add', 'Edit', 'Delete', 'Update', 'Cancel',
'Search'];
            this.orderidrules = { required: true, number: true };
            this.customeridrules = { required: true };
            this.filterSettings = { type: 'Excel' };
        }
        public onLoad(args: any): void {
            (this.treegrid as TreeGridComponent).grid.adaptiveDlgTarget =
document.getElementsByClassName('e-mobile-content')[0] as HTMLElement;
        }
    }

```

MAIN.TS

```

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

```

Please refer to our [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to learn how to present and manipulate data.

Loading animation in Angular Treegrid component

The tree grid has an option to show a loading indicator in-between the time of fetching the data and binding it to the tree grid during initial rendering or refreshing or after performing any tree grid action like sorting, paging and more. The tree grid supports two indicator types, which is achieved by setting the `loadingIndicator.indicatorType` property to `Spinner` or `Shimmer`. The default value of the indicator type is `"Spinner"`.

In the following sample, the Shimmer indicator is displayed while the tree grid is loading and refreshing when using the remote data.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit } from '@angular/core';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [treeColumnIndex]='1'
  hasChildMapping='isParent' parentIdMapping='ParentItem' idMapping='TaskID'
  [allowSorting]="true" [allowPaging]="true">
    <e-columns>
      <e-column field='TaskID' headerText='Task ID' width='90'
  textAlign='Right'></e-column>
      <e-column field='TaskName' headerText='Task Name'
  width='170'></e-column>
      <e-column field='StartDate' headerText='Start Date' width='130'
  format="yMd" textAlign='Right'></e-column>
      <e-column field='Duration' headerText='Duration' width='80'
  textAlign='Right'></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: DataManager;
  public loadingIndicator?: any;
  public pageSettings?: Object;
  ngOnInit(): void {
    this.data = new DataManager({
      url: 'https://ej2services.syncfusion.com/production/web-
  services/api/SelfReferenceData',
      adaptor: new WebApiAdaptor, crossDomain: true
    })
  }
}
```

```

    });
    this.loadingIndicator = { indicatorType: 'Shimmer' };
    this.pageSettings = { pageCount: 3 };
  }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

State Persistence

State persistence in Angular Treegrid component

State persistence refers to the TreeGrid's state maintained in the browser's [localStorage](#) even if the browser is refreshed or if you move to the next page within the browser.

State persistence stores treegrid's model object in the local storage when the [enablePersistence](#) is defined as true.

You can refer to our [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Get or set local storage value in Angular Treegrid component

If the [enablePersistence](#) property is set to true, the treegrid property value is saved in the `window.localStorage` for reference. You can get/set the localStorage value by using the `getItem/setItem` method in the `window.localStorage`.

`typescript

//get the TreeGrid model.

```
let value: string = window.localStorage.getItem('treegridTreeGrid'); // "treegridTreeGrid" is component name + component id.
```

```
let model: Object = JSON.parse(model);
```

,

`typescript

//set the TreeGrid model.

```
window.localStorage.setItem('treegridTreeGrid', JSON.stringify(model)); // "treegridTreeGrid" is component name + component id.
```

,

Pdf Export

Pdf export in Angular Treegrid component

PDF export allows exporting TreeGrid data to PDF document. You need to use the [pdfExport](#) method for exporting. To enable PDF export in the treegrid, set the [allowPdfExport](#) as true.

To use PDF export, inject the [Link to the Video](#) module in treegrid.

You can check this video to learn about how to perform Exporting and its customization in Angular TreeGrid.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, PdfExportService, ToolbarService } from
 '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { PdfExportProperties } from '@syncfusion/ej2-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    PdfExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowPdfExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pager?: Object;
  @ViewChild('treegrid')
```

```

public treeGridObj?: TreeGridComponent;
public toolbarOptions?: ToolbarItems[];
ngOnInit(): void {
    this.data = sampleData;
    this.pager = { pageSize: 7 };
    this.toolbarOptions = ['PdfExport'];
}
toolbarClick(args: Object | any) : void {
    if (args['item'].text === 'PDF Export') {
        (this.treeGridObj as TreeGridComponent).pdfExport();
    }
}
}

```

MAIN.TS

```

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

```

<!-- Multiple exporting

PDF export provides an option for exporting multiple treegrids to same file. In this exported document, each treegrid will be exported to new page of document in same file.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, PdfExportService, ToolbarService } from
'@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData, sampleData } from './datasource';
import { ToolbarItems, TreeGridComponent } from '@syncfusion/ej2-angular-
treegrid';
@Component({
    imports: [

        TreeGridModule,
        ButtonModule,
        DropDownListAllModule,
    ],
    providers: [PageService,
        PdfExportService,
        ToolbarService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowPdfExport]='true' (pageSettings)='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
        <e-columns>

```

```

        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
    </ejs-treegrid>
    <br>
    <ejs-treegrid [dataSource]='data' height='220'
parentIdMapping='parentID' idMapping='TaskID' [allowPaging]='true'
[treeColumnIndex]='1'>
        <e-columns>
            <e-column field='TaskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
            <e-column field='TaskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
            <e-column field='StartDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
            <e-column field='Duration' headerText='Duration'
textAlign='Right' width=110></e-column>
        </e-columns>
    </ejs-treegrid>
    ,
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public toolbarOptions?: ToolbarItems[];
        @ViewChild('treegrid')
        public treeGridObj?: TreeGridComponent;
        pager?: { pageSize: number; };
        firstTreeGrid?: TreeGridComponent;
        secondTreeGrid?: TreeGridComponent;
        ngOnInit(): void {
            this.data = projectData;
            this.pager = { pageSize: 7 };
            this.toolbarOptions = ['PdfExport'];
        }
        toolbarClick(args: Object | any) : void {
            if (args['item'].text === 'PDF Export') {
                let firstGridPdfExport: Promise<Object> =
this.firstTreeGrid?.pdfExport({}, true) as any;
                firstGridPdfExport.then((pdfData: Object) => {
                    this.secondTreeGrid?.pdfExport({}, false, pdfData);
                });
            }
        }
    }
}

```

MAIN.TS

```

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

```

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

Show spinner while exporting

Show or hide the spinner component while exporting the tree grid using the `showSpinner` or `hideSpinner` methods. Use the [toolbarClick](#) event to show spinner before exporting and hide a spinner in the [pdfExportComplete](#) event after the export.

In the following demo, the default spinner component is rendered when exporting the tree grid.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, PdfExportService, ToolbarService } from
 '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { TreeGridComponent, ToolbarItems } from '@syncfusion/ej2-angular-
treegrid';
import { ClickEventArgs } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    PdfExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid id='TreeGrid' [dataSource]='data'
height='220' (toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowPdfExport]='true' [treeColumnIndex]='1' childMapping='subtasks'
[toolbar]='toolbarOptions' (pdfExportComplete)='pdfExportComplete()'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: object[];
  public toolbarOptions?: ToolbarItems[];
  @ViewChild('treegrid') public treegrid?: TreeGridComponent;
```



```

ngOnInit(): void {
    this.data = sampleData;
    this.toolbarOptions = ['PdfExport'];
}
toolbarClick(args: ClickEventArgs): void {
    if (args['item'].text === 'PDF Export') {
        (this.treegrid as TreeGridComponent).showSpinner();
        (this.treegrid as TreeGridComponent).pdfExport();
    }
}
pdfExportComplete(): void {
    (this.treegrid as TreeGridComponent).hideSpinner();
}
}

```

MAIN.TS

```

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

```

Custom data source

PDF export provides an option to define datasource dynamically before exporting. To export data dynamically, define the `dataSource` in `exportProperties`

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, PdfExportService, ToolbarService } from
 '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { PdfExportProperties, RowDataBoundEventArgs,
 PdfQueryCellInfoEventArgs } from '@syncfusion/ej2-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    PdfExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'

```

```

[allowPdfExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
</ejs-treegrid>`
}))
export class AppComponent implements OnInit {
    public data?: Object[];
    public pager?: Object;
    @ViewChild('treegrid')
    public treeGridObj?: TreeGridComponent;
    public toolbarOptions?: ToolbarItems[];
    ngOnInit(): void {
        this.data = sampleData;
        this.pager = { pageSize: 7 };
        this.toolbarOptions = ['PdfExport'];
    }
    toolbarClick(args: Object | any) : void {
        if (args['item'].text === 'PDF Export') {
            let exportProperties: PdfExportProperties = {
                dataSource: sampleData,
            };
            (this.treeGridObj as
TreeGridComponent).pdfExport(exportProperties);
        }
    }
}
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Pdf export options in Angular Treegrid component

Export current page

PDF export provides an option to export the current page into PDF. To export current page, define the exportType to `CurrentPage`.

APP.COMPONENT.TS

```

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

```

```

import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, PdfExportService, ToolbarService } from
 '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { PdfTrueTypeFont } from '@syncfusion/ej2-pdf-export';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { PdfExportProperties } from '@syncfusion/ej2-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    PdfExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowPdfExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
    </ejs-treegrid>`
  })
export class AppComponent implements OnInit {
  public data?: Object[];
  public pager?: Object;
  public toolbarOptions?: ToolbarItems[];
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
    this.pager = { pageSize: 7 };
    this.toolbarOptions = ['PdfExport'];
  }
  toolbarClick(args: Object | any) : void {
    if (args['item'].text === 'PDF Export') {
      let exportProperties: PdfExportProperties = {
        exportType: 'CurrentPage'
      };
      (this.treeGridObj as
TreeGridComponent).pdfExport(exportProperties);
    }
  }
}

```

```

    }
  }
}

```

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 selected records only

Export the selected records data by passing it to the [exportProperties.dataSource](#) Property in the [toolbarClick](#) event.

In the following demo, get the selected records using the [getSelectedRecords](#) method and pass the selected data to the [PdfExport](#) property.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, PdfExportService, ToolbarService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import {
  TreeGridComponent, ToolbarItems, ToolbarService, PdfExportService,
  PageService, SelectionSettingsModel
} from '@syncfusion/ej2-angular-treegrid';
import { ClickEventArgs } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    PdfExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid id='TreeGrid' [dataSource]='data'
height='220' (toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowPdfExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
[selectionSettings]='selectionSettings' childMapping='subtasks'
[toolbar]='toolbarOptions'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
    </e-columns>
  `
})
export class AppComponent implements OnInit {
  constructor(private treeGrid: TreeGridComponent, private toolbarService: ToolbarService, private pdfExportService: PdfExportService, private pageService: PageService) {}

  ngOnInit(): void {
    this.toolbarService.toolbarItems = [
      { text: 'Export', click: this.exportData }
    ];
  }

  exportData(): void {
    const selectedRecords = this.treeGrid.getSelectedRecords();
    this.pdfExportService.dataSource = selectedRecords;
    this.pdfExportService.export();
  }
}

```

```

        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
</ejs-treegrid>`,
    providers: [ToolbarService, PdfExportService, PageService]
})
export class AppComponent implements OnInit {
    public data?: object[];
    public toolbarOptions?: ToolbarItems[];
    public selectionSettings?: SelectionSettingsModel;
    public initialPage?: object;
    @ViewChild('treegrid') public treegrid?: TreeGridComponent;
    pager: any;
    ngOnInit(): void {
        this.data = sampleData;
        this.toolbarOptions = ['PdfExport'];
        this.selectionSettings = { type: 'Multiple' };
    }
    toolbarClick(args: ClickEventArgs) {
        if (args.item.id === 'TreeGrid_gridcontrol_pdfexport') {
            const selectedRecords = (this.treegrid as
TreeGridComponent).getSelectedRecords();
            const exportProperties = {
                dataSource: selectedRecords
            };
            this.treegrid?.pdfExport(exportProperties);
        }
    }
}

```

MAIN.TS

```

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

```

Export hidden columns

PDF export provides an option to export hidden columns of the TreeGrid by defining the `includeHiddenColumn` as `true`.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, PdfExportService, ToolbarService } from
'@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { PdfTrueTypeFont } from '@syncfusion/ej2-pdf-export';

```

```

import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { PdfExportProperties } from '@syncfusion/ej2-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    PdfExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowPdfExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
[visible]='false' textAlign='Right' width=110></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pager?: Object;
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  public toolbarOptions?: ToolbarItems[];
  ngOnInit(): void {
    this.data = sampleData;
    this.pager = { pageSize: 7 };
    this.toolbarOptions = ['PdfExport'];
  }
  toolbarClick(args: Object | any) : void {
    if (args['item'].text === 'PDF Export') {
      let exportProperties: PdfExportProperties = {
        includeHiddenColumn: true
      };
      this.treeGridObj?.pdfExport(exportProperties);
    }
  }
}

```

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 columns on exported PDF

You can show a hidden column or hide a visible column while exporting the treegrid using [toolbarClick](#) and [pdfExportComplete](#) events.

In the `toolbarClick` event, based on `args.item.text` as `PDF Export`. We can show or hide columns by setting `column.visible` property to `true` or `false` respectively.

In the `pdfExportComplete` event, We have reversed the state back to the previous state.

In the below example, we have `Duration` as a hidden column in the treegrid. While exporting, we have changed `Duration` to visible column and `StartDate` as hidden column.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, PdfExportService, ToolbarService } from
 '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { PdfTrueTypeFont } from '@syncfusion/ej2-pdf-export';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { PdfExportProperties } from '@syncfusion/ej2-grids';
import { Column, TreeGridComponent } from '@syncfusion/ej2-angular-
treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    PdfExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(pdfExportComplete)='pdfExportComplete($event)'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowPdfExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
```

```

        <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
</ejs-treegrid>`
}))
export class AppComponent implements OnInit {
    public data?: Object[];
    public pager?: Object;
    @ViewChild('treegrid')
    public treeGridObj?: TreeGridComponent;
    public toolbarOptions?: ToolbarItems[];
    ngOnInit(): void {
        this.data = sampleData;
        this.pager = { pageSize: 7 };
        this.toolbarOptions = ['PdfExport'];
    }
    toolbarClick(args: Object | any) : void {
        if (args['item'].text === 'PDF Export') {
            let cols: Column[] = this.treeGridObj?.grid.columns as Column[];
            cols[2].visible = false;
            cols[3].visible = true;
            this.treeGridObj?.pdfExport();
        }
    }
    pdfExportComplete(args: any): void {
        let cols: Column[] = this.treeGridObj?.grid.columns as Column[];
        cols[3].visible = false;
        cols[2].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));

```

[How to change page orientation](#)

Page orientation can be changed Landscape(Default Portrait) for the exported document using the `exportProperties`.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, PdfExportService, ToolbarService } from
'@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { PdfTrueTypeFont } from '@syncfusion/ej2-pdf-export';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';

```



```

import { PdfExportProperties } from '@syncfusion/ej2-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    PdfExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowPdfExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pager?: Object;
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  public toolbarOptions?: ToolbarItems[];
  ngOnInit(): void {
    this.data = sampleData;
    this.pager = { pageSize: 7 };
    this.toolbarOptions = ['PdfExport'];
  }
  toolbarClick(args: Object | any) : void {
    if (args['item'].text === 'PDF Export') {
      let exportProperties: PdfExportProperties = {
        pageOrientation: 'Landscape'
      };
      (this.treeGridObj as
TreeGridComponent).pdfExport(exportProperties);
    }
  }
}

```

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 change page size

Page size can be customized for the exported document using the `exportProperties`.

Supported page sizes are:

- Letter
- Note
- Legal
- A0
- A1
- A2
- A3
- A5
- A6
- A7
- A8
- A9
- B0
- B1
- B2
- B3
- B4
- B5
- Archa
- Archb
- Archc
- Archd
- Arche
- Flsa
- HalfLetter
- Letter11x17
- Ledger

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, PdfExportService, ToolbarService } from
 '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { PdfTrueTypeFont } from '@syncfusion/ej2-pdf-export';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { PdfExportProperties } from '@syncfusion/ej2-grids';
```

```

import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    PdfExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowPdfExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pager?: Object;
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  public toolbarOptions?: ToolbarItems[];
  ngOnInit(): void {
    this.data = sampleData;
    this.pager = { pageSize: 7 };
    this.toolbarOptions = ['PdfExport'];
  }
  toolbarClick(args: Object | any) : void {
    if (args['item'].text === 'PDF Export') {
      let exportProperties: PdfExportProperties = {
        pageSize: 'Letter'
      };
      (this.treeGridObj as
TreeGridComponent).pdfExport(exportProperties);
    }
  }
}

```

MAIN.TS

```

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

```

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

To customize PDF export

PDF export provides an option to customize mapping of treegrid to exported PDF document.

File name for exported document

You can assign the file name for the exported document by defining `fileName` property in `PdfExportProperties`.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, PdfExportService, ToolbarService } from
 '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { PdfExportProperties } from '@syncfusion/ej2-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    PdfExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowPdfExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pager?: Object;
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
```

```

public toolbarOptions?: ToolbarItems[];
ngOnInit(): void {
    this.data = sampleData;
    this.pager = { pageSize: 7 };
    this.toolbarOptions = ['PdfExport'];
}
toolbarClick(args: Object | any) : void {
    if (args['item'].text === 'PDF Export') {
        let exportProperties: PdfExportProperties = {
            fileName: "new.pdf"
        };
        (this.treeGridObj as
TreeGridComponent).pdfExport(exportProperties);
    }
}
}

```

MAIN.TS

```

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

```

Default fonts for PDF exporting

By default, treegrid uses Helvetica font in the exported document. You can change the default font by using pdfExportProperties.theme property. The available default fonts are,

- Helvetica
- TimesRoman
- Courier
- Symbol
- ZapfDingbats

The code example for changing default font,

`typescript

```

import { PdfStandardFont, PdfFontFamily, PdfFontStyle } from '@syncfusion/ej2-pdf-export';
...
let pdfExportProperties: PdfExportProperties = {
    theme: {
        header: {font: new PdfStandardFont(PdfFontFamily.TimesRoman, 11, PdfFontStyle.Bold),
        record: { font: new PdfStandardFont(PdfFontFamily.TimesRoman, 10) }
    }
};

```

Add custom font for PDF exporting

You can change the default font of TreeGrid header, content and caption cells in the exported document by using `pdfExportProperties.theme` property.

In the following example, we have used Advent Pro font to export the treegrid with Hungarian fonts.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, PdfExportService, ToolbarService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData, adventProFont } from './datasource';
import { PdfTrueTypeFont } from '@syncfusion/ej2-pdf-export';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { PdfExportProperties } from '@syncfusion/ej2-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    PdfExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowPdfExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pager?: Object;
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  public toolbarOptions?: ToolbarItems[];
  ngOnInit(): void {
    this.data = sampleData;
  }
}
```

```

        this.pager = { pageSize: 7 };
        this.toolbarOptions = ['PdfExport'];
    }
    toolbarClick(args: Object | any) : void {
        if (args['item'].text === 'PDF Export') {
            let exportProperties: PdfExportProperties = {
                theme: {
                    header: {font: new PdfTrueTypeFont(adventProFont,
12) },
                    record: { font: new PdfTrueTypeFont(adventProFont,
9) }
                }
            };
            (this.treeGridObj as
TreeGridComponent).pdfExport(exportProperties);
        }
    }
}

```

MAIN.TS

```

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

```

PdfTrueTypeFont accepts base 64 format of the Custom Font.

Pdf cell style customization in Angular Treegrid component

Conditional cell formatting

TreeGrid cells in the exported PDF can be customized or formatted using [pdfQueryCellInfo](#) event. In this event, we can format the treegrid cells of exported PDF document based on the column cell value.

In the below sample, we have set the background color for **Duration** column in the exported document by **args.cell** and **backgroundColor** property.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, PdfExportService, ToolbarService } from
 '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { PdfTrueTypeFont } from '@syncfusion/ej2-pdf-export';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { PdfExportProperties, RowDataBoundEventArgs,
PdfQueryCellInfoEventArgs } from '@syncfusion/ej2-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
    imports: [

```

```

        TreeGridModule,
        ButtonModule,
        DropDownListAllModule,
    ],
    providers: [PageService,
                PdfExportService,
                ToolbarService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(queryCellInfo)='queryCellInfo($event)'
(pdfQueryCellInfo)='pdfQueryCellInfo($event)'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowPdfExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
        <e-columns>
            <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
            <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
            <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
            <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
        </e-columns>
    </ejs-treegrid>`
))
export class AppComponent implements OnInit {
    public data?: Object[];
    public pager?: Object;
    @ViewChild('treegrid')
    public treeGridObj?: TreeGridComponent;
    public toolbarOptions?: ToolbarItems[];
    ngOnInit(): void {
        this.data = sampleData;
        this.pager = { pageSize: 7 };
        this.toolbarOptions = ['PdfExport'];
    }
    toolbarClick(args: Object | any) : void {
        if (args['item'].text === 'PDF Export') {
            (this.treeGridObj as TreeGridComponent).pdfExport();
        }
    }
    pdfQueryCellInfo(args: PdfQueryCellInfoEventArgs | any): void {
        if (args.column.field === 'duration') {
            if (+args.value === 0 || args.value === "") {
                args.style = { backgroundColor: '#336c12' };
            }
            else if (args.value < 3) {
                args.style = { backgroundColor: '#7b2b1d' };
            }
        }
    }
    queryCellInfo(args: RowDataBoundEventArgs | any): void {
        if (args.data['duration'] === 0 && args.column.field === 'duration' )
        {
            args.cell.style.background= '#336c12';

```



```

    } else if (args.data['duration'] < 3 && args.column.field ===
'duration') {
        args.cell.style.background= '#7b2b1d';
    }
}
}

```

MAIN.TS

```

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

```

Theme

PDF export provides an option to include theme for exported PDF document.

To apply theme in exported PDF, define the **theme** in **exportProperties**.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, PdfExportService, ToolbarService } from
 '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { PdfExportProperties, RowDataBoundEventArgs,
PdfQueryCellInfoEventArgs } from '@syncfusion/ej2-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    PdfExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowPdfExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>

```

```

        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
</ejs-treegrid>`
}))
export class AppComponent implements OnInit {
    public data?: Object[];
    public pager?: Object;
    @ViewChild('treegrid')
    public treeGridObj?: TreeGridComponent;
    public toolbarOptions?: ToolbarItems[];
    ngOnInit(): void {
        this.data = sampleData;
        this.pager = { pageSize: 7 };
        this.toolbarOptions = ['PdfExport'];
    }
    toolbarClick(args: Object | any) : void {
        if (args['item'].text === 'PDF Export') {
            let exportProperties: PdfExportProperties = {
                theme: {
                    header: {
                        fontColor: '#64FA50', fontName: 'Calibri', fontSize:
17, bold: true, border: { color: '#64FA50', lineStyle: 'Thin' }
                    },
                    record: {
                        fontColor: '#64FA50', fontName: 'Calibri', fontSize:
17, bold: true
                    }
                }
            };
            (this.treeGridObj as
TreeGridComponent).pdfExport(exportProperties);
        }
    }
}

```

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, material theme is applied to exported PDF document.

Adding header and footer in Angular Treegrid component

You can customize text, page number, line, page size and changing orientation in header and footer.

How to write a text in header or footer

You can add text either in Header or Footer of exported PDF document.

`typescript

```
let exportProperties: PdfExportProperties = {
```

```

header: {
  fromTop: 0,
  height: 130,
  contents: [
    {
      type: 'Text',
      value: "Task Details",
      position: { x: 0, y: 50 },
      style: { textBrushColor: '#000000', fontSize: 13 }
    },
  ]
}
`

```

How to draw a line in header or footer

you can add line either in Header or Footer of the exported PDF document.

Supported line styles:

- dash
- dot
- dashdot
- dashdotdot
- solid

```

`typescript
let exportProperties: PdfExportProperties = {
  header: {
    fromTop: 0,
    height: 130,
    contents: [
      {
        type: 'Line',
        style: { penColor: '#000080', penSize: 2, dashStyle: 'Solid' },
        points: { x1: 0, y1: 4, x2: 685, y2: 4 }
      }
    ]
  }
}
`

```

```
}
,
```

Add page number in header or footer

you can add page number either in Header or Footer of exported PDF document.

Supported page number types:

- LowerLatin - a, b, c,
- UpperLatin - A, B, C,
- LowerRoman - i, ii, iii,
- UpperRoman - I, II, III,
- Number - 1,2,3.

```
`typescript
let exportProperties: PdfExportProperties = {
  header: {
    fromTop: 0,
    height: 130,
    contents: [
      {
        type: 'PageNumber',
        pageNumberType: 'Arabic',
        format: 'Page { $current } of { $total }', //optional
        position: { x: 0, y: 25 },
        style: { textBrushColor: '#ffff80', fontSize: 15, hAlign: 'Center' }
      }
    ]
  }
},
```

Insert an image in header or footer

Image (Base64 string) can be added in the exported document in header/footer using the `exportProperties`.

```
`typescript
let exportProperties: PdfExportProperties = {
  header: {
    fromTop: 0,
```

```

height: 130,
contents: [
{
type: 'Image',
src: image,
position: { x: 40, y: 10 },
size: { height: 100, width: 250 },
}
]
}
}
,

```

The below code illustrates the pdf export customization.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, PdfExportService, ToolbarService } from
 '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { PdfTrueTypeFont } from '@syncfusion/ej2-pdf-export';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { PdfExportProperties } from '@syncfusion/ej2-grids';
import { image } from './image';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
],
providers: [PageService,
    PdfExportService,
    ToolbarService],
standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid [dataSource]='data' height='220' #treegrid
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowPdfExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>

```

```

        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
</ejs-treegrid>`
})
export class AppComponent implements OnInit {
    public data?: Object[];
    public pager?: Object;
    @ViewChild('treegrid')
    public treeGridObj?: TreeGridComponent;
    public toolbarOptions?: ToolbarItems[];
    ngOnInit(): void {
        this.data = sampleData;
        this.pager = { pageSize: 7 };
        this.toolbarOptions = ['PdfExport'];
    }
    toolbarClick(args: Object | any) : void {
        if (args['item'].text === 'PDF Export') {
            let pdfExportProperties: PdfExportProperties = {
                header: {
                    fromTop: 0,
                    height: 130,
                    contents: [
                        {
                            type: 'Image',
                            src: image,
                            position: { x: 40, y: 10 },
                            size: { height: 100, width: 250 },
                        }
                    ]
                },
                footer: {
                    fromBottom: 160,
                    height: 150,
                    contents: [
                        {
                            type: 'PageNumber',
                            pageNumberType: 'Arabic',
                            format: 'Page {$current} of {$total}',
                            position: { x: 0, y: 25 },
                            style: { textBrushColor: '#ffff80', fontSize: 15

```

MAIN.TS

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

Exporting tree grid in server in Angular Tree Grid component

The Tree Grid have an option to export the data to PDF in server side using tree grid server export library.

Server dependencies

The Server side export functionality is shipped in the Syncfusion.EJ2.TreeGridExport package, which is available in Essential Studio and [nuget.org](https://www.nuget.org). The following list of dependencies is required for tree grid server side PDF exporting action.

- Syncfusion.EJ2
- Syncfusion.EJ2.TreeGridExport

Server configuration

The following code snippet shows server configuration using ASP.NET Core Controller Action.

To Export the tree grid in server side, You need to call the [serverPdfExport](#) method for passing the tree grid properties to server exporting action.

`typescript

```
public IActionResult ServerSideExporting()
{
    var order = TreeData.GetDefaultData();
    ViewBag.dataSource = order;
    return View();
}

public IActionResult PdfExport(string treeGridModel)
{
    if (treeGridModel == null)
    {
        return View();
    }
    TreeGridExcelExport exp = new TreeGridExcelExport();
    Syncfusion.EJ2.TreeGrid.TreeGrid gridProperty = ConvertTreeGridObject(treeGridModel);
    return exp.ExportToPdf<TreeData>(gridProperty, TreeData.GetDefaultData());
}
```

```

private Syncfusion.EJ2.TreeGrid.TreeGrid ConvertTreeGridObject(string gridProperty)
{
    Syncfusion.EJ2.TreeGrid.TreeGrid TreeGridModel =
    (Syncfusion.EJ2.TreeGrid.TreeGrid)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
    typeof(Syncfusion.EJ2.TreeGrid.TreeGrid));

    TreeGridColumnModel cols =
    (TreeGridColumnModel)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
    typeof(TreeGridColumnModel));

    TreeGridModel.Columns = cols.columns;

    return TreeGridModel;
}

public class TreeGridColumnModel
{
    public List<TreeGridColumn> columns { get; set; }
}
`
`typescript
import { Component, OnInit, ViewChild } from '@angular/core';
import { ToolbarItems, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
@Component({
    selector: 'app-root',
    template: `<ejs-treegrid #treegrid id='TreeGrid' parentIdMapping='ParentItem' [dataSource]='data'
    [toolbar]='toolbar' height='273px'(toolbarClick)='toolbarClick($event)'>
    <e-columns>
    <e-column field='TaskID' headerText='Task ID' width='90' textAlign='Right'></e-column>
    <e-column field='TaskName' headerText='Task Name' width='170'></e-column>
    <e-column field='StartDate' headerText='Start Date' width='130' format='yMd' textAlign='Right'></e-
    column>
    <e-column field='Duration' headerText='Duration' width='80' textAlign='Right'></e-column>
    </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {

```



```

public data: DataManager;
public toolbar: ToolbarItems[];
public dataManager: DataManager = new DataManager({
url: 'Home/UrlDatasource',
adaptor: new UrlAdaptor()
});
@ViewChild('treegrid')
public treegrid: TreeGridComponent;
ngOnInit(): void {
this.data = this.dataManager;
this.toolbar = ['PdfExport'];
}
toolbarClick(args: ClickEventArgs): void {
if (args.item.id === 'TreeGridgridcontrolpdfexport') { // 'TreeGridpdfexport' -> TreeGrid component id +
+ toolbar item name
this.treegrid.serverPdfExport('Home/PdfExport');
}
}
}
,

```

Rotate a header text to a certain degree in the exported tree grid on the server side

The Tree Grid has support to customize the column header styles such as changing text orientation, the font color, and so on in the exported PDF file. To achieve this requirement, define the **BeginCellLayout** event of the **PdfExportProperties** with an event handler to perform the required action.

The **PdfHeaderCellRendering** will be triggered when creating a column header for the pdf document to be exported. Collect the column header details in this event and handle the custom in the **BeginCellLayout** event handler.

In the following demo, the **DrawString** method from the **Graphics** is used to rotate the header text of the column header inside the **BeginCellLayout** event handler.

A PDF exporting is not supported to rotate the column header on the client side.

```

`typescript
public IActionResult PdfExport(string treeGridModel)
{
if (treeGridModel == null)
{

```

```
return View();
}
TreeGridPdfExport exp = new TreeGridPdfExport();
TreeGrid gridProperty = ConvertTreeGridObject(treeGridModel);
gridProperty.PdfHeaderCellRendering = PdfHeaderQueryCellInfo;
PdfGrid grid = new PdfGrid();
Syncfusion.EJ2.TreeGridExport.PdfExportProperties pdfExportProperties = new
Syncfusion.EJ2.TreeGridExport.PdfExportProperties();
pdfExportProperties.IsRepeatHeader = true;
exp.BeginCellLayout = new PdfGridBeginCellLayoutEventHandler(BeginCellEvent);
System.Collections.IEnumerable data = Syncfusion.EJ2.Base.Dat.DataTableToJson(ViewBag.dataSource);
var result = exp.PdfExport<dynamic>(gridProperty, data, pdfExportProperties);
return View();
}
public void BeginCellEvent(object sender, PdfGridBeginCellLayoutEventArgs args)
{
    PdfGrid grid = (PdfGrid)sender;
    var brush = new PdfSolidBrush(new PdfColor(Color.DimGray));
    args.Graphics.Save();
    args.Graphics.TranslateTransform(args.Bounds.X + 50, args.Bounds.Height + 40); // give the value for
    bounds x and Y by the user
    args.Graphics.RotateTransform(-60); // give the rotate degree value by the user
    // Draw the text at particular bounds.
    args.Graphics.DrawString(headerValues[args.CellIndex], new PdfStandardFont(PdfFontFamily.Helvetica,
    10), brush, new PointF(0, 0));
    if (args.IsHeaderRow)
    {
        grid.Headers[0].Cells[args.CellIndex].Value = string.Empty;
    }
    args.Graphics.Restore();
}
private void PdfHeaderQueryCellInfo(object pdf)
{

```

```

Syncfusion.EJ2.TreeGridExport.PdfHeaderCellEventArgs name =
(Syncfusion.EJ2.TreeGridExport.PdfHeaderCellEventArgs)pdf;
PdfGrid grid = new PdfGrid();
headerValues.Add(name.Column.HeaderText);
var longestString = headerValues.Where(s => s.Length == headerValues.Max(m => m.Length)).First();
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 6);
SizeF size = font.MeasureString(longestString);
name.Headers[0].Height = size.Width * 2;
}
`

```

Excel Export

Excel export in Angular Treegrid component

The excel export allows exporting TreeGrid data to Excel document. You need to use the [excelExport](#) method for exporting. To enable Excel export in the treegrid, set the [allowExcelExport](#) as true.

To use excel export, You need to inject the [Link to the Video](#) module in treegrid.

You can check this video to learn about how to perform Exporting and its customization in Angular TreeGrid.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, ExcelExportService, ToolbarService } from
 '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { ExcelExportProperties } from '@syncfusion/ej2-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,

  ],
  providers: [PageService,
    ExcelExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowExcelExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>

```

```

        <e-columns>
            <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
            <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
            <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
            <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
        </e-columns>
    </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public pager?: Object;
        @ViewChild('treegrid')
        public treeGridObj?: TreeGridComponent;
        public toolbarOptions?: ToolbarItems[];
        ngOnInit(): void {
            this.data = sampleData;
            this.pager = { pageSize: 7 };
            this.toolbarOptions = ['ExcelExport'];
        }
        toolbarClick(args: Object | any) : void {
            if (args['item'].text === 'Excel Export') {
                this.treeGridObj?.excelExport();
            }
        }
    }
}

```

MAIN.TS

```

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

```

Persist collapsed state

You can persist the collapsed state in the exported document by defining `isCollapsedStatePersist` property as true in `TreeGridExcelExportProperties` parameter of [excelExport](#) method.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, ExcelExportService, ToolbarService } from
'@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems, TreeGridExcelExportProperties } from
'@syncfusion/ej2-treegrid';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';

```

```

@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    ExcelExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowExcelExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pager?: Object;
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  public toolbarOptions?: ToolbarItems[];
  ngOnInit(): void {
    this.data = sampleData;
    this.pager = { pageSize: 7 };
    this.toolbarOptions = ['ExcelExport'];
  }
  toolbarClick(args: Object | any) : void {
    if (args['item'].text === 'Excel Export') {
      let excelExportProperties: TreeGridExcelExportProperties = {
        isCollapsedStatePersist: true
      };
      this.treeGridObj?.excelExport(excelExportProperties);
    }
  }
}

```

MAIN.TS

```

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

```

The CSV export allows exporting tree grid data to text document. Use the [csvExport](#) method for exporting. To enable CSV export in the treegrid, set the [allowCsvExport](#) as true.

Custom data source

The excel export provides an option to define datasource dynamically before exporting. To export data dynamically, define the `dataSource` in `exportProperties`.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, ExcelExportService, ToolbarService } from
 '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { ExcelExportProperties } from '@syncfusion/ej2-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    ExcelExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowExcelExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pager?: Object;
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  public toolbarOptions?: ToolbarItems[];
  ngOnInit(): void {
    this.data = sampleData;
```

```

        this.pager = { pageSize: 7 };
        this.toolbarOptions = ['ExcelExport'];
    }
    toolbarClick(args: Object | any) : void {
        if (args['item'].text === 'Excel Export') {
            let excelExportProperties: ExcelExportProperties = {
                dataSource: sampleData
            };
            this.treeGridObj?.excelExport(excelExportProperties);
        }
    }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Excel export options in Angular Treegrid component

To customize excel export

The excel export provides an option to customize mapping of the treegrid to excel document.

Export hidden columns

Excel Export provides an option to export hidden columns of TreeGrid by defining the `includeHiddenColumn` as `true`.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, ExcelExportService, ToolbarService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { ExcelExportProperties } from '@syncfusion/ej2-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    ExcelExportService,
    ToolbarService],
})

```

```

standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowExcelExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
[visible]='false' textAlign='Right' width=110></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pager?: Object;
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  public toolbarOptions?: ToolbarItems[];
  ngOnInit(): void {
    this.data = sampleData;
    this.pager = { pageSize: 7 };
    this.toolbarOptions = ['ExcelExport'];
  }
  toolbarClick(args: Object | any) : void {
    if (args['item'].text === 'Excel Export') {
      let exportProperties: ExcelExportProperties = {
        includeHiddenColumn: true
      };
      (this.treeGridObj as
TreeGridComponent).excelExport(exportProperties);
    }
  }
}

```

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 columns on exported excel

You can show a hidden column or hide a visible column while printing the treegrid using [toolbarClick](#) and [excelExportComplete](#) events.

In the `toolbarClick` event, based on `args.item.text` as `Excel Export`. We can show or hide columns by setting `column.visible` property to `true` or `false` respectively.

In the `excelExportComplete` event, We have reversed the state back to the previous state.

In the below example, we have **Duration** as a hidden column in the treegrid. While exporting, we have changed **Duration** to visible column and **StartDate** as hidden column.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, ExcelExportService, ToolbarService } from
 '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { Column, TreeGridComponent } from '@syncfusion/ej2-angular-
treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    ExcelExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(excelExportComplete)='excelExportComplete($event)'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowExcelExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pager?: Object;
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  public toolbarOptions?: ToolbarItems[];
  ngOnInit(): void {
    this.data = sampleData;
    this.pager = { pageSize: 7 };
    this.toolbarOptions = ['ExcelExport'];
  }
}
```

```

toolbarClick(args: Object | any) : void {
    if (args['item'].text === 'Excel Export') {
        let cols: Column[] = this.treeGridObj?.grid.columns as Column[];
        cols[2].visible = false;
        cols[3].visible = true;
        this.treeGridObj?.excelExport();
    }
}
excelExportComplete(ars: any): void {
    let cols: Column[] = this.treeGridObj?.grid.columns as Column[];
    cols[3].visible = false;
    cols[2].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));

```

[File name for exported document](#)

You can assign the file name for the exported document by defining `fileName` property in [ExcelExportProperties](#).

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, ExcelExportService, ToolbarService } from
 '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { ExcelExportProperties } from '@syncfusion/ej2-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    ExcelExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'

```

```

[allowExcelExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
</ejs-treegrid>`
})
export class AppComponent implements OnInit {
    public data?: Object[];
    public pager?: Object;
    @ViewChild('treegrid')
    public treeGridObj?: TreeGridComponent;
    public toolbarOptions?: ToolbarItems[];
    ngOnInit(): void {
        this.data = sampleData;
        this.pager = { pageSize: 7 };
        this.toolbarOptions = ['ExcelExport'];
    }
    toolbarClick(args: Object | any) : void {
        if (args['item'].text === 'Excel Export') {
            let excelExportProperties: ExcelExportProperties = {
                fileName:"new.xlsx"
            };
            this.treeGridObj?.excelExport(excelExportProperties);
        }
    }
}
}

```

MAIN.TS

```

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

```

Excel cell style customization in Angular Treegrid component

Conditional cell formatting

TreeGrid cells in the exported Excel can be customized or formatted using [excelQueryCellInfo](#) event. In this event, we can format the treegrid cells of exported Excel document based on the column cell value.

In the below sample, we have set the background color for **Duration** column in the exported excel by **args.cell** and **backgroundColor** property.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'

```

```

import { PageService, ExcelExportService, ToolbarService } from
 '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { ExcelExportProperties, RowDataBoundEventArgs,
ExcelQueryCellInfoEventArgs } from '@syncfusion/ej2-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    ExcelExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(queryCellInfo)='queryCellInfo($event)'
(excelQueryCellInfo)='excelQueryCellInfo($event)'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowExcelExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pager?: Object;
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  public toolbarOptions?: ToolbarItems[];
  ngOnInit(): void {
    this.data = sampleData;
    this.pager = { pageSize: 7 };
    this.toolbarOptions = ['ExcelExport'];
  }
  toolbarClick(args: Object | any) : void {
    if (args['item'].text === 'Excel Export') {
      (this.treeGridObj as TreeGridComponent).excelExport();
    }
  }
  excelQueryCellInfo(args: ExcelQueryCellInfoEventArgs | any): void {

```

```

        if(args.column.field == 'duration'){
            if(args.value === 0 || args.value === "") {
                args.style = {backColor: '#336c12'};
            }
            else if(args.value < 3) {
                args.style = {backColor: '#7b2b1d'};
            }
        }
    }
    queryCellInfo(args: RowDataBoundEventArgs | any): void {
        if (args.data['duration'] == 0 && args.column.field === 'duration' )
    {
        args.cell.style.background= '#336c12';
    } else if (args.data['duration'] < 3 && args.column.field ===
'duration') {
        args.cell.style.background= '#7b2b1d';
    }
    }
}
}

```

MAIN.TS

```

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

```

Theme

The excel export provides an option to include theme for exported excel document.

To apply theme in exported Excel, define the `theme` in `exportProperties`.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, ExcelExportService, ToolbarService } from
 '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { ExcelExportProperties } from '@syncfusion/ej2-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    ExcelExportService,
    ToolbarService],

```

```

standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
(toolbarClick)='toolbarClick($event)' [allowPaging]='true'
[allowExcelExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
childMapping='subtasks' [toolbar]='toolbarOptions'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pager?: Object;
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  public toolbarOptions?: ToolbarItems[];
  ngOnInit(): void {
    this.data = sampleData;
    this.pager = { pageSize: 7 };
    this.toolbarOptions = ['ExcelExport'];
  }
  toolbarClick(args: Object | any) : void {
    if (args['item'].text === 'Excel Export') {
      let exportProperties: ExcelExportProperties = {
        theme: {
          header: {
            fontColor: '#64FA50', fontName: 'Calibri', fontSize:
17, bold: true, borders: { color: '#64FA50', lineStyle: 'Thin' } as any
          },
          record: {
            fontColor: '#64FA50', fontName: 'Calibri', fontSize:
17, bold: true
          }
        }
      };
      (this.treeGridObj as
TreeGridComponent).excelExport(exportProperties);
    }
  }
}

```

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, material theme is applied to exported excel document.

Adding header and footer in Angular Treegrid component

The excel export provides an option to include header and footer content for exported excel document.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, ExcelExportService, ToolbarService } from
 '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { ExcelExportProperties } from '@syncfusion/ej2-grids';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    ExcelExportService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `

```

```

    }
    toolbarClick(args: Object | any) : void {
        if (args['item'].text === 'Excel Export') {
            let excelExportProperties: ExcelExportProperties = {
                header: {
                    headerRows: 7,
                    rows: [
                        { cells: [{ colSpan: 4, value: "Northwind Traders",
style: { fontColor: '#C67878', fontSize: 20, hAlign: 'Center', bold: true, }
}] },
                        { cells: [{ colSpan: 4, value: "2501 Aerial Center
Parkway", style: { fontColor: '#C67878', fontSize: 15, hAlign: 'Center',
bold: true, } } ] },
                        { cells: [{ colSpan: 4, value: "Suite 200
Morrisville, NC 27560 USA", style: { fontColor: '#C67878', fontSize: 15,
hAlign: 'Center', bold: true, } } ] },
                        { cells: [{ colSpan: 4, value: "Tel +1 888.936.8638
Fax +1 919.573.0306", style: { fontColor: '#C67878', fontSize: 15, hAlign:
'Center', bold: true, } } ] },
                        { cells: [{ colSpan: 4, hyperlink: { target:
'https://www.northwind.com/', displayText: 'www.northwind.com' }, style: {
hAlign: 'Center' } } ] },
                        { cells: [{ colSpan: 4, hyperlink: { target:
'mailto:support@northwind.com' }, style: { hAlign: 'Center' } } ] },
                    ]
                },
                footer: {
                    footerRows: 4,
                    rows: [
                        { cells: [{ colSpan: 4, value: "Thank you for your
business!", style: { hAlign: 'Center', bold: true } } ] },
                        { cells: [{ colSpan: 4, value: "!Visit Again!",
style: { hAlign: 'Center', bold: true } } ] }
                    ]
                },
            };
            (this.treeGridObj as
TreeGridComponent).excelExport(excelExportProperties);
        }
    }
}

```

MAIN.TS

```

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

```

Exporting tree grid in server in Angular Tree Grid component

The Tree Grid have an option to export the data to Excel in server side using tree grid server export library.

Server dependencies

The Server side export functionality is shipped in the Syncfusion.EJ2.TreeGridExport package, which is available in Essential Studio and [nuget.org](https://www.nuget.org). The following list of dependencies is required for tree grid server side Excel exporting action.

- Syncfusion.EJ2
- Syncfusion.EJ2.TreeGridExport

Server configuration

The following code snippet shows server configuration using ASP.NET Core Controller Action.

To Export the tree grid in server side, You need to call the [serverExcelExport](#) method for passing the tree grid properties to server exporting action.

```
`typescript
public IActionResult ServerSideExporting()
{
    var order = TreeData.GetDefaultData();
    ViewBag.dataSource = order;
    return View();
}

public IActionResult ExcelExport(string treeGridModel)
{
    if (treeGridModel == null)
    {
        return View();
    }

    TreeGridExcelExport exp = new TreeGridExcelExport();
    Syncfusion.EJ2.TreeGrid.TreeGrid gridProperty = ConvertTreeGridObject(treeGridModel);
    return exp.ExportToExcel<TreeData>(gridProperty, TreeData.GetDefaultData());
}

private Syncfusion.EJ2.TreeGrid.TreeGrid ConvertTreeGridObject(string gridProperty)
{
    Syncfusion.EJ2.TreeGrid.TreeGrid TreeGridModel =
    (Syncfusion.EJ2.TreeGrid.TreeGrid)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
    typeof(Syncfusion.EJ2.TreeGrid.TreeGrid));

    TreeGridColumnModel cols =
    (TreeGridColumnModel)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
    typeof(TreeGridColumnModel));
}
```

```

TreeGridModel.Columns = cols.columns;
return TreeGridModel;
}
public class TreeGridColumnModel
{
public List<TreeGridColumn> columns { get; set; }
}
`
`typescript
import { Component, OnInit, ViewChild } from '@angular/core';
import { ToolbarItems, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
@Component({
selector: 'app-root',
template: `<ejs-treegrid #treegrid id='TreeGrid' parentIdMapping='ParentItem' [dataSource]='data'
[toolbar]='toolbar' height='273px'(toolbarClick)='toolbarClick($event)'>
<e-columns>
<e-column field='TaskID' headerText='Task ID' width='90' textAlign='Right'></e-column>
<e-column field='TaskName' headerText='Task Name' width='170'></e-column>
<e-column field='StartDate' headerText='Start Date' width='130' format='yMd' textAlign='Right'></e-
column>
<e-column field='Duration' headerText='Duration' width='80' textAlign='Right'></e-column>
</e-columns>
</ejs-treegrid>`
})
export class AppComponent implements OnInit {
public data: DataManager;
public toolbar: ToolbarItems[];
public dataManager: DataManager = new DataManager({
url: 'Home/UrlDatasource',
adaptor: new UrlAdaptor()
});
@ViewChild('treegrid')

```

```

public treegrid: TreeGridComponent;
ngOnInit(): void {
  this.data = this.dataManager;
  this.toolbar = ['ExcelExport'];
}
toolbarClick(args: ClickEventArgs): void {
  if (args.item.id === 'TreeGridgridcontrolexcelexport') { // 'TreeGridexcelexport' -> TreeGrid component id
    + + toolbar item name
    this.treegrid.serverExcelExport('Home/ExcelExport');
  }
}
}
,

```

CSV Export in server side

You can export the tree grid to CSV format by using the [serverCsvExport](#) method which will pass the tree grid properties to server.

In the below demo, we have invoked the above method inside the [toolbarClick](#) event. In server side, we have deserialized the tree grid properties and passed to the [ExportToCsv](#) method which will export the properties to CSV format.

```

`typescript
public IActionResult ServerSideExporting()
{
  var order = TreeData.GetDefaultData();
  ViewBag.dataSource = order;
  return View();
}

public IActionResult CsvExport(string treeGridModel)
{
  if (treeGridModel == null)
  {
    return View();
  }

  TreeGridExcelExport exp = new TreeGridExcelExport();
  Syncfusion.EJ2.TreeGrid.TreeGrid gridProperty = ConvertTreeGridObject(treeGridModel);
  return exp.ExportToCsv<TreeData>(gridProperty, TreeData.GetDefaultData());
}

```

```

}
private Syncfusion.EJ2.TreeGrid.TreeGrid ConvertTreeGridObject(string gridProperty)
{
    Syncfusion.EJ2.TreeGrid.TreeGrid TreeGridModel =
    (Syncfusion.EJ2.TreeGrid.TreeGrid)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
    typeof(Syncfusion.EJ2.TreeGrid.TreeGrid));

    TreeGridColumnModel cols =
    (TreeGridColumnModel)Newtonsoft.Json.JsonConvert.DeserializeObject(gridProperty,
    typeof(TreeGridColumnModel));

    TreeGridModel.Columns = cols.columns;
    return TreeGridModel;
}

public class TreeGridColumnModel
{
    public List<TreeGridColumn> columns { get; set; }
}
`
`typescript
import { Component, OnInit, ViewChild } from '@angular/core';
import { ToolbarItems, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { DataManager, UrlAdaptor } from '@syncfusion/ej2-data';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';

@Component({
    selector: 'app-root',
    template: `<ejs-treegrid #treegrid id='TreeGrid' parentIdMapping='ParentItem' [dataSource]='data'
    [toolbar]='toolbar' height='273px'(toolbarClick)='toolbarClick($event)'>
    <e-columns>
    <e-column field='TaskID' headerText='Task ID' width='90' textAlign='Right'></e-column>
    <e-column field='TaskName' headerText='Task Name' width='170'></e-column>
    <e-column field='StartDate' headerText='Start Date' width='130' format='yMd' textAlign='Right'></e-
    column>
    <e-column field='Duration' headerText='Duration' width='80' textAlign='Right'></e-column>
    </e-columns>
    </ejs-treegrid>`
})

```

```

export class AppComponent implements OnInit {
  public data: DataManager;
  public toolbar: ToolbarItems[];
  public dataManager: DataManager = new DataManager({
    url: 'Home/UrlDatasource',
    adaptor: new UrlAdaptor()
  });
  @ViewChild('treegrid')
  public treegrid: TreeGridComponent;
  ngOnInit(): void {
    this.data = this.dataManager;
    this.toolbar = ['CsvExport'];
  }
  toolbarClick(args: ClickEventArgs): void {
    if (args.item.id === 'TreeGridgridcontrolcsvexport') { // 'TreeGridcsvexport' -> Tree Grid component id +
      + toolbar item name
      this.treegrid.serverCsvExport('Home/CsvExport');
    }
  }
}

```

Rotate a header text to a certain degree in the exported grid on the server side

The TreeGrid has support to customize the column header styles such as changing text orientation, the font color, and so on in the exported Excel file. To achieve this requirement, use the `ExcelHeaderCellRendering` event of the tree grid.

The `ExcelHeaderCellRendering` will be triggered when creating a column header for the excel document to be exported in the server side. Customize the column header in this event.

In the following demo, using the `HeaderCellRotate` method of the `TreeGridExcelExport` class in the `ExcelHeaderCellRendering` event, you can rotate the header text of the column header in the excel exported document.

```

`typescript
public IActionResult ExcelExport(string treeGridModel)
{
  if (treeGridModel == null)
  {

```

```

return View();
}
TreeGridExcelExport exp = new TreeGridExcelExport();
Syncfusion.EJ2.TreeGrid.TreeGrid gridProperty = ConvertTreeGridObject(treeGridModel);
gridProperty.ExcelHeaderCellRendering = ExcelHeaderQueryCellInfo;
return (ActionResult)exp.ExcelExport<TreeGridItems>(gridProperty, TreeGridItems.GetDefaultData());
}
private void ExcelHeaderQueryCellInfo(object excel)
{
Syncfusion.EJ2.TreeGridExport.ExcelHeaderCellEventArgs name =
(Syncfusion.EJ2.TreeGridExport.ExcelHeaderCellEventArgs)excel;
List<string> headerValues = new List<string>();
headerValues.Add(name.Column.HeaderText);
var longestString = headerValues.Where(s => s.Length == headerValues.Max(m => m.Length)).First();
TreeGridExcelExport exp = new TreeGridExcelExport();
var size = exp.ExcelTextSize(name.Style.Font.FontName, (float)name.Style.Font.Size, longestString);
name.Cell.RowHeight = size.Width;
exp.HeaderCellRotate(name, 45); // Give the rotate degree value by the user.
name.Style.Borders.LineStyle = Syncfusion.XlsIO.ExcelLineStyle.None;
}
,

```

Context menu in Angular Treegrid component

The TreeGrid has options to show the context menu when right clicked on it. To enable this feature, you need to define either default or custom item in the [contextMenuItems](#).

To use the context menu, inject the **ContextMenu** module in the treegrid.

The default items are in the following table.

Items	Description
AutoFit	Auto fit the current column.
AutoFitAll	Auto fit all columns.
Edit	Edit the current record.
Delete	Delete the current record.
Save	Save the edited record.
Cancel	Cancel the edited state.

PdfExport | Export the treegrid data as Pdf document.

ExcelExport | Export the treegrid data as Excel document.

CsvExport | Export the treegrid data as CSV document.

SortAscending | Sort the current column in ascending order.

SortDescending | Sort the current column in descending order.

FirstPage | Go to the first page.

PrevPage | Go to the previous page.

LastPage | Go to the last page.

NextPage | Go to the next page.

AddRow | Add new row to the treegrid.

Indent | Indents the record to one level of hierarchy.

Outdent | Outdents the record to one level of hierarchy.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService, EditService, ToolbarService,
    ResizeService, ExcelExportService, PdfExportService, ContextMenuService
} from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit } from '@angular/core';
import { sampleData } from '../datasource';
import { ToolbarItems, RowDD } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    SortService,
    FilterService,
    EditService,
    SortService, ResizeService,
    ExcelExportService,
    PdfExportService, ContextMenuService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
[allowPaging]='true' pageSettings='pager'
[contextMenuItems]='contextMenuItems'
[allowResizing]='true' [allowSorting]='true' [treeColumnIndex]='1'
childMapping='subtasks' [allowExcelExport]='true' [allowPdfExport]='true'>`
})
```

```

        <e-columns>
            <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
            <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
            <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
            <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
        </e-columns>
    </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public pager?: Object;
        public editSettings?: Object;
        public contextMenuItems?: Object[];
        ngOnInit(): void {
            this.data = sampleData;
            this.editSettings = {allowEditing: true, allowAdding: true,
allowDeleting: true, mode:"Row"};
            this.contextMenuItems = ['AutoFit', 'AutoFitAll', 'SortAscending',
'SortDescending', 'Edit', 'Delete', 'Save', 'Cancel', 'PdfExport',
'ExcelExport', 'CsvExport', 'FirstPage', 'PrevPage', 'LastPage', 'NextPage',
'Indent', 'Outdent'];
            this.pager = { pageSize: 8 }
        }
    }
}

```

MAIN.TS

```

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

```

Custom context menu items

The custom context menu items can be added by defining the [contextMenuItems](#) as a collection of [contextMenuItemModel](#).

Actions for this customized items can be defined in the [contextMenuClick](#) event.

In the below sample, we have shown context menu item for parent rows to expand or collapse child rows.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService, EditService, ToolbarService,
ResizeService, ExcelExportService, PdfExportService, ContextMenuService
} from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';

```



```

import { sampleData } from './datasource';
import { getValue, isNullOrUndefined } from '@syncfusion/ej2-base';
import { BeforeOpenCloseEventArgs } from '@syncfusion/ej2-inputs';
import { MenuEventArgs } from '@syncfusion/ej2-navigations';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,

  ],
  providers: [PageService,
    SortService,
    FilterService,
    EditService,
    SortService, ResizeService,
    ExcelExportService,
    PdfExportService, ContextMenuService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
[allowPaging]='true' pageSettings='pager'
[contextMenuItems]='contextMenuItems' [treeColumnIndex]='1'
(contextMenuClick)='contextMenuClick($event)'
(contextMenuOpen)='contextMenuOpen($event)' childMapping='subtasks'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pager?: Object;
  public editSettings?: Object;
  public contextMenuItems?: Object[];
  @ViewChild('treegrid')
  public treeGridObj?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
    this.editSettings = {allowEditing: true, allowAdding: true,
allowDeleting: true, mode:"Row"};
    this.contextMenuItems = [
      {text: 'Collapse the Row', target: '.e-content', id:
'collapserow'},
      {text: 'Expand the Row', target: '.e-content', id:
'expandrow'}
    ];
    this.pager = { pageSize: 8 }
  }
}

```

```

    }
    contextMenuClick(args?: MenuEventArgs): void {
        (this.treeGridObj as TreeGridComponent).getColumnByField('taskID');
        if ((args as MenuEventArgs).item.id === 'collapserow') {
            (this.treeGridObj as TreeGridComponent).collapseRow(<HTMLTableRowElement>((this.treeGridObj as TreeGridComponent).getSelectedRows()[0]));
        } else {
            (this.treeGridObj as TreeGridComponent).expandRow(<HTMLTableRowElement>((this.treeGridObj as TreeGridComponent).getSelectedRows()[0]));
        }
    }
    contextMenuOpen(arg?: BeforeOpenCloseEventArgs) : void {
        let elem: Element = (arg as BeforeOpenCloseEventArgs).event.target as Element;
        let uid: string = (elem.closest('.e-row') as Element).getAttribute('data-uid') as string;
        if (isNullOrUndefined(getValue('hasChildRecords', (this.treeGridObj as TreeGridComponent).grid.getRowObjectFromUID(uid).data))) {
            (arg as BeforeOpenCloseEventArgs).cancel = true;
        } else {
            let flag: boolean = getValue('expanded', (this.treeGridObj as TreeGridComponent).grid.getRowObjectFromUID(uid).data);
            let val: string = flag ? 'none' : 'block';

            document.querySelectorAll('li#expandrow')[0].setAttribute('style', 'display: ' + val + ';');
            val = !flag ? 'none' : 'block';

            document.querySelectorAll('li#collapserow')[0].setAttribute('style', 'display: ' + val + ';');
        }
    }
}

```

MAIN.TS

```

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

```

Enable and disable context menu items dynamically

You can enable and disable the context menu items using the [enableItems](#) method in [contextMenuOpen](#) event.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService, EditService, ToolbarService,
    ResizeService, ExcelExportService, PdfExportService, ContextMenuService
} from '@syncfusion/ej2-angular-treegrid'

```

```

import {ButtonModule} from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { getValue, isNullOrUndefined } from '@syncfusion/ej2-base';
import { BeforeOpenCloseEventArgs } from '@syncfusion/ej2-inputs';
import { MenuEventArgs } from '@syncfusion/ej2-navigations';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    SortService,
    FilterService,
    EditService,
    SortService, ResizeService,
    ExcelExportService,
    PdfExportService, ContextMenuService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' #treegrid height='220'
[allowPaging]='true' pageSettings='pager' [editSettings]='editSettings'
[contextMenuItems]='contextMenuItems' [treeColumnIndex]='1'
(contextMenuOpen)='contextMenuOpen($event)'
(contextMenuClick)='contextMenuClick($event)' childMapping='subtasks'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
      <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
      <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
      <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public pager?: Object;
  public editSettings?: Object;
  public contextMenuItems?: Object[];
  @ViewChild('treegrid')
  public treegrid?: TreeGridComponent;
  ngOnInit(): void {
    this.data = sampleData;
    this.editSettings = {allowEditing: true, allowAdding: true,
allowDeleting: true, mode:"Row"};
    this.contextMenuItems=[
      { text: 'Edit Record', target: '.e-content', id: 'Edit_record' },
      { text: 'Delete Record', target: '.e-content', id: 'Delete_record'
    },
  ],

```

```

    ];
    this.pager = { pageSize: 8 }
  }
  contextMenuClick(args?: MenuEventArgs): void {
    if((args as MenuEventArgs).element.innerHTML == "Edit Record"){
      (this.treegrid as TreeGridComponent).startEdit((args as
MenuEventArgs | any).rowInfo.row);
    }
    else if((args as MenuEventArgs).element.innerHTML == "Delete
Record"){
      (this.treegrid as TreeGridComponent).deleteRecord((args as
MenuEventArgs | any).rowInfo.row);
    }
  }
  contextMenuOpen(args?: BeforeOpenCloseEventArgs): void {
    if ((args as BeforeOpenCloseEventArgs |
any).rowInfo.rowData.hasChildRecords == true){
      (this.treegrid as
TreeGridComponent).grid.contextMenuModule.contextMenu.enableItems(['Edit
Record'], true); //Enable edit
      (this.treegrid as
TreeGridComponent).grid.contextMenuModule.contextMenu.enableItems(['Delete
Record'], false); //Disable delete
    } else {
      (this.treegrid as
TreeGridComponent).grid.contextMenuModule.contextMenu.enableItems(['Edit
Record'], false); //Disable edit
      (this.treegrid as
TreeGridComponent).grid.contextMenuModule.contextMenu.enableItems(['Delete
Record'], true); //Enable edit
    }
  }
}

```

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 hide or show an item in context menu for specific area inside of treegrid by defining the [target](#) property.

You can refer to our [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Accessibility in Angular Treegrid component

The Tree Grid 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 Tree Grid 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 Tree Grid component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Tree Grid component:

| Attributes | Purpose |

| --- | --- |

| `role=treegrid` | Used to convey a significant and contextual message to the user. |

| `aria-selected` | Accurately reflect the selection state, whether it's single-select or multi-select. |

| **aria-expanded** | It can be used to show whether a node is expanded or collapsed, making it easier for screen reader users to navigate and understand the hierarchy. |

| **aria-sort** | Indicate the current sorting order of a table column for users with disabilities, facilitating accessible data presentation and interaction. |

| **aria-busy** | Loading state to improve accessibility for users, particularly those relying on screen readers. |

| **aria-invalid** | To indicate whether the user's input in a form field is valid or invalid, aiding users, including those with disabilities, in understanding and correcting their input. |

| **aria-grabbed** | Provides accessibility information for users interacting with draggable elements |

| **aria-owns** | Establishing relationships between an element and the elements it owns or controls. |

| **aria-label** | Provides an accessible name for the close icon. |

Keyboard interaction

The Tree Grid 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 Tree Grid component.

Interaction Keys | Description

PageDown | Goes to the next page.

PageUp | Goes to the previous page.

Ctrl + Alt + PageDown | Goes to the last page.

Ctrl + Alt + PageUp | Goes to the first page.

Alt + PageDown | Goes to the next page.

Alt + PageUp | Goes to the previous page.

Home | Goes to the first cell.

End | Goes to the last cell.

Ctrl + Home | Goes to the first row.

Ctrl + End | Goes to the last row.

DownArrow | Moves the cell focus downward.

UpArrow | Moves the cell focus upward.

LeftArrow | Moves the cell focus left side.

RightArrow | Moves the cell focus right side.

Shift + DownArrow | Extends the row/cell selection downwards.

Shift + UpArrow | Extends the row/cell selection upwards.

Shift + LeftArrow | Extends the cell selection to the left side.

Shift + RightArrow | Extends the cell selection to the right side.

Enter | Moves the row/cell selection downward. If current cell is in edit state, then completes the editing. If the current cell is a header then performs sorting.

Shift + Enter | Moves the row/cell selection upward. If the current cell is a header then clears sorting for the selected column.

Ctrl + Enter | If the current cell is a header then performs multi-sorting.

Tab | Moves the cell selection right side.

Shift + Tab | Moves the cell selection left side.

Esc | Deselects all the rows/cells.

Ctrl + A | Selects all the rows/cells.

UpArrow | Moves up a row/cell selection.

DownArrow | Moves down a row/cell selection.

RightArrow | Moves to the right cell selection.

LeftArrow | Moves to the left cell selection.

Ctrl + Shift + DownArrow | Expands the selected group.

Ctrl + DownArrow | Expands all the visible groups.

Ctrl + Shift + UpArrow | Collapses the selected group.

Ctrl + UpArrow | Collapses all the visible groups.

Ctrl + P | Prints the TreeGrid.

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion Angular components](#)

You can refer to our [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Clipboard in Angular Treegrid component

The clipboard provides an option to copy selected rows or cells data into the clipboard.

The following list of keyboard shortcuts is supported in the Tree Grid to copy selected rows or cells data into clipboard.

Interaction keys | Description

Ctrl + C | Copy selected rows or cells data into clipboard.

Ctrl + Shift + H | Copy selected rows or cells data with header into clipboard.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
import { SelectionSettingsModel } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid [dataSource]='data' [allowSelection]='true'
[allowPaging]='true' height='260' [selectionSettings]='selectionOptions'
[pageSettings]='pageSettings' childMapping='subtasks' [treeColumnIndex]='1'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
      <e-column field='taskName' headerText='Task Name'
width='200'></e-column>
      <e-column field='startDate' headerText='Start Date' width='100'
format='yMd' textAlign='Right'></e-column>
      <e-column field='duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
      <e-column field='progress' headerText='Progress' width='90'
textAlign='Right'></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public selectionOptions?: SelectionSettingsModel;
  public pageSettings?: Object;
  ngOnInit(): void {
    this.data = sampleData;
    this.selectionOptions = { type: 'Multiple' };
    this.pageSettings = { pageSize: 10 };
  }
}
```

MAIN.TS

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


Copy to clipboard by external buttons

To copy selected rows or cells data into clipboard with help of external buttons, you need to invoke the [copy](#) method.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewEncapsulation, ViewChild } from '@angular/core';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
import { sampleData } from './datasource';
import { TreeGridComponent, SelectionSettingsModel } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService],
  standalone: true,
  selector: 'app-container',
  template:
    `<button ej-button id='copy' (click)='copy()'>Copy</button>
    <button ej-button id='copyHeader'
    (click)='copyHeader()'>CopyHeader</button>
    <ejs-treegrid #treegrid [dataSource]='data' [allowSelection]='true'
    [allowPaging]='true' height='230' [selectionSettings]='selectionOptions'
    [pageSettings]='pageSettings' childMapping='subtasks' [treeColumnIndex]='1'>
      <e-columns>
        <e-column field='taskID' headerText='Task ID' width='70'
        textAlign='Right'></e-column>
        <e-column field='taskName' headerText='Task Name'
        width='200'></e-column>
        <e-column field='startDate' headerText='Start Date' width='100'
        format='yMd' textAlign='Right'></e-column>
        <e-column field='duration' headerText='Duration' width='90'
        textAlign='Right'></e-column>
        <e-column field='progress' headerText='Progress' width='90'
        textAlign='Right'></e-column>
      </e-columns>
    </ejs-treegrid>`,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public selectionOptions?: SelectionSettingsModel;
  public pageSettings?: Object;
  @ViewChild('treegrid')
```

```

public treeGridObj?: TreeGridComponent;
ngOnInit(): void {
    this.data = sampleData;
    this.selectionOptions = { type: 'Multiple' };
    this.pageSettings = { pageSize: 10 };
}
copy() {
    (this.treeGridObj as TreeGridComponent).copy();
}
copyHeader() {
    (this.treeGridObj as TreeGridComponent).copy(true);
}
}

```

MAIN.TS

```

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

```

Copy Hierarchy Modes

Tree Grid provides support for a set of copy modes with [copyHierarchyMode](#) property. The below are the type of filter mode available in TreeGrid.

- **Parent** : This is the default copy hierarchy mode in Tree Grid. Clipboard value have the selected records with its parent records. If the selected records not have any parent record then the selected record will be in clipboard.
- **Child** : Clipboard value have the selected records with its child record. If the selected records do not have any child record then the selected records will be in clipboard.
- **Both** : Clipboard value have the selected records with its both parent and child record. If the selected records do not have any parent and child record then the selected records alone in clipboard.
- **None** : Only the Selected records will be in clipboard.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
import { sampleData } from './datasource';
import { ChangeEventArgs } from '@syncfusion/ej2-angular-dropdowns';
import { SelectionSettingsModel, TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

```

```

        TreeGridModule,
        ButtonModule,
        DropDownListAllModule
    ],
    providers: [PageService],
    standalone: true,
    selector: 'app-container',
    template:
`<div style="padding-top: 7px; float:left">Hierarchy Mode</div><div
style="padding-left: 10px; display: inline-block"><ejs-dropdownlist
(change)='onChange($event)' [dataSource]='dropData' value='Parent'
[fields]='fields'></ejs-dropdownlist></div>
    <ejs-treegrid #treegrid [dataSource]='data' [allowSelection]='true'
[allowPaging]='true' height='230' copyHierarchyMode='Parent'
[selectionSettings]='selectionOptions' [pageSettings]='pageSettings'
childMapping='subtasks' [treeColumnIndex]='1'>
        <e-columns>
            <e-column field='taskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
            <e-column field='taskName' headerText='Task Name'
width='200'></e-column>
            <e-column field='startDate' headerText='Start Date' width='100'
format="yMd" textAlign='Right'></e-column>
            <e-column field='duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
            <e-column field='progress' headerText='Progress' width='90'
textAlign='Right'></e-column>
        </e-columns>
    </ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public dropData?: Object[];
        public fields?: Object;
        public selectionOptions?: SelectionSettingsModel;
        public pageSettings?: Object ;
        @ViewChild('treegrid')
        public treeGridObj?: TreeGridComponent;
        ngOnInit(): void {
            this.data = sampleData;
            this.dropData = [
                { id: 'Parent', mode: 'Parent' },
                { id: 'Child', mode: 'Child' },
                { id: 'Both', mode: 'Both' },
                { id: 'None', mode: 'None' },
            ];
            this.fields = { text: 'mode', value: 'id' };
            this.selectionOptions = { type: 'Multiple' };
            this.pageSettings = { pageSize: 10 };
        }
        onChange(e: ChangeEventArgs): any {
            let mode: any = <string>e.value;
            (this.treeGridObj as TreeGridComponent).copyHierarchyMode = mode;
        }
    }

```

MAIN.TS

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

Limitations of Copy Functionality

- Only current view records will be available in copy clipboard.

AutoFill

AutoFill Feature allows you to copy the data of selected cells and paste it to another cells by just dragging the autofill icon of the selected cells up to required cells. This feature is enabled by defining `enableAutoFill` property as true.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService, EditService, ToolbarService }
  from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
import { SelectionSettingsModel, EditSettingsModel, ToolbarItems } from
  '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    SortService,
    FilterService,
    EditService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data'
[enableAutoFill]='true' [enableHover]='false' [allowPaging]='true'
[pageSettings]='pageSettings' [editSettings]='editSettings'
[allowSelection]='true' [toolbar]='toolbar'
[selectionSettings]='selectionOptions' height='220' childMapping='subtasks'
[treeColumnIndex]='1'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
      <e-column field='taskName' headerText='Task Name'
width='200'></e-column>
```

```

        <e-column field='startDate' headerText='Start Date'
width='100' format='yMd' textAlign='Right'></e-column>
        <e-column field='duration' headerText='Duration'
width='90' textAlign='Right'></e-column>
        <e-column field='progress' headerText='Progress'
width='90' textAlign='Right'></e-column>
    </e-columns>
</ejs-treegrid>`
    })
    export class AppComponent implements OnInit {
        public data?: Object[];
        public selectionOptions?: SelectionSettingsModel;
        public editSettings?: EditSettingsModel;
        public pageSettings?: Object ;
        public toolbar?: ToolbarItems[];
        ngOnInit(): void {
            this.data = sampleData;
            this.selectionOptions = { type: 'Multiple', mode: 'Cell',
cellSelectionMode: 'Box' };
            this.editSettings= { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Batch' },
            this.toolbar = ['Add', 'Update', 'Cancel'];
            this.pageSettings = {pageSize: 10};
        }
    }
}

```

MAIN.TS

```

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

```

* If `enableAutoFill` is set to true, then the autofill icon will be displayed on cell selection to copy cells.

* It requires the selection `mode` to be `Cell`, `cellSelectionMode` to be `Box` and also Batch Editing should be enabled.

Limitations of AutoFill

- Since the string values are not parsed to number and date type, so when the selected string type cells are dragged to number type cells then it will display as **NaN**. For date type cells, when the selected string type cells are dragged to date type cells then it will display as an **empty cell**.
- Linear series and the sequential data generations are not supported in this autofill feature.

Paste

You can able to copy the content of a cell or a group of cells by selecting the cells and pressing Ctrl + C shortcut key and paste it to another set of cells by selecting the cells and pressing Ctrl + V shortcut key.

Paste the cell programmatically by using the [Paste](#) method in the tree grid. To use, you need to pass the data to paste the row index and column index as parameters to the method.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService, EditService, ToolbarService }
  from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit } from '@angular/core';
import { sampleData } from './datasource';
import { SelectionSettingsModel, EditSettingsModel, ToolbarItems,
  PageSettingsModel } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule,
  ],
  providers: [PageService,
    SortService,
    FilterService,
    EditService,
    ToolbarService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegrid [dataSource]='data' height='220'
[enableHover]='false' [allowPaging]='true' [pageSettings]='pageSettings'
[editSettings]='editSettings' [toolbar]='toolbar' [allowSelection]='true'
[selectionSettings]='selectionOptions' childMapping='subtasks'
[treeColumnIndex]='1'>
    <e-columns>
      <e-column field='taskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
      <e-column field='taskName' headerText='Task Name'
width='200'></e-column>
      <e-column field='startDate' headerText='Start Date'
width='100' format='yMd' textAlign='Right'></e-column>
      <e-column field='duration' headerText='Duration'
width='90' textAlign='Right'></e-column>
      <e-column field='progress' headerText='Progress'
width='90' textAlign='Right'></e-column>
    </e-columns>
  </ejs-treegrid>`
})
export class AppComponent implements OnInit {
  public data?: Object[];
  public selectionOptions?: SelectionSettingsModel;
  public editSettings?: EditSettingsModel;
  public pageSettings?: PageSettingsModel;
  public toolbar?: ToolbarItems[];
  ngOnInit(): void {
    this.data = sampleData;
    this.selectionOptions = { type: 'Multiple', mode: 'Cell',
cellSelectionMode: 'Box' };
  }
}

```

```

    this.editSettings= { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Batch' },
    this.toolbar = ['Add', 'Update', 'Cancel'];
  }
}

```

MAIN.TS

```

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

```

To perform paste functionality, it requires the selection **mode** to be **Cell**, **cellSelectionMode** to be **Box** and also Batch Editing should be enabled.

Limitations of Paste Functionality

- Since the string values are not parsed to number and date type, so when the copied string type cells are pasted to number type cells then it will display as **NaN**. For date type cells, when the copied string format cells are pasted to date type cells then it will display as an **empty cell**.

You can refer to our [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Global local in Angular Treegrid component

Localization

The [Localization](#) library allows you to localize default text content of the TreeGrid. The treegrid component has static text on some features (like toolbar area text, filter menu text, pager information text, etc.) 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 treegrid.

Locale keywords | Text

EmptyRecord | No records to display

True | true

False | false

ExpandAll | Expand All

CollapseAll | Collapse All

RowIndent | Indent

RowOutdent | Outdent

InvalidFilterMessage | Invalid Filter Data

FilterbarTitle | \s filter bar cell

Add | Add

Edit | Edit

Cancel | Cancel

Update | Update

Delete | Delete

Print | Print

Pdfexport | PDF Export

Excelexport | Excel Export

Wordexport | Word Export

Csvexport | CSV Export

Search | Search

Save | Save

EditOperationAlert | No records selected for edit operation

DeleteOperationAlert | No records selected for delete operation

SaveButton | Save

OKButton | OK

CancelButton | Cancel

EditFormTitle | Details of

AddFormTitle | Add New Record

ConfirmDelete | Are you sure you want to Delete Record?

SearchColumns | search columns

Matches | No Matches Found

FilterButton | Filter

ClearButton | Clear

StartsWith | Starts With

EndsWith | Ends With

Contains | Contains

Equal | Equal

NotEqual | Not Equal

LessThan | Less Than

LessThanOrEqual | Less Than Or Equal

GreaterThan | Greater Than

GreaterThanOrEqual | Greater Than Or Equal

ChooseDate | Choose a Date

EnterValue | Enter the value

autoFitAll | Auto Fit all columns

autoFit | Auto Fit this column

Export | Export

FirstPage | First Page

LastPage | Last Page

PreviousPage | Previous Page

NextPage | Next Page

SortAscending | Sort Ascending

SortDescending | Sort Descending

EditRecord | Edit Record

DeleteRecord | Delete Record

Above | Above

Below | Below

AddRow | Add Row

FilterMenu | Filter

SelectAll | Select All

Blanks | Blanks

FilterTrue | True

FilterFalse | False

NoResult | No Matches Found

ClearFilter | Clear Filter

NumberFilter | Number Filters

TextFilter | Text Filters

DateFilter | Date Filters

MatchCase | Match Case

Between | Between

CustomFilter | Custom Filter

CustomFilterPlaceholder | Enter the value

CustomFilterDatePlaceholder | Choose a date

AND | AND

OR | OR

ShowRowsWhere | Show rows where:

currentPageInfo | {0} of {1} pages

totalItemsInfo | ({0} items)

firstPageTooltip | Go to first page

lastPageTooltip | Go to last page

nextPageTooltip | Go to next page

previousPageTooltip | Go to previous page

nextPagerTooltip | Go to next pager

previousPagerTooltip | Go to previous pager

pagerDropDown | Items per page

pagerAllDropDown | Items

All | All

Loading translations

To load translation object in an application, use `load` function of the `L10n` class.

The following example demonstrates the TreeGrid in `Deutsch` culture.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { L10n } from '@syncfusion/ej2-base';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';

L10n.load({
  'de-DE': {
    'treegrid': {
      'EmptyRecord': 'Keine Aufzeichnungen angezeigt',
      'Expand All': 'Alle erweitern',
      'Collapse All': 'Alles einklappen',
      'Print': 'Drucken',
      'Pdfexport': 'PDF-Export',
      'Excelext': 'Excel-Export',
      'Wordexport': 'Word-Export',
      'FilterButton': 'Filter',
      'ClearButton': 'klar',
      'StartsWith': 'Beginnt mit',
      'EndsWith': 'Endet mit',
      'Contains': 'Enthält',
      'Equal': 'Gleich',
      'NotEqual': 'Nicht gleich',
      'LessThan': 'Weniger als',
      'LessThanOrEqual': 'Weniger als oder gleich',
      'GreaterThan': 'Größer als',
      'GreaterThanOrEqual': 'Größer als oder gleich',
    }
  }
});
```

```

        "EnterValue": "Geben Sie den Wert ein",
        "FilterMenu": "Filter"
    },
    'pager': {
        'currentPageInfo': '{0} von {1} Seiten',
        'totalItemsInfo': '({0} Beiträge)',
        'firstPageTooltip': 'Zur ersten Seite',
        'lastPageTooltip': 'Zur letzten Seite',
        'nextPageTooltip': 'Zur nächsten Seite',
        'previousPageTooltip': 'Zurück zur letzten Seit',
        'nextPagerTooltip': 'Zum nächsten Pager',
        'previousPagerTooltip': 'Zum vorherigen Pager'
    },
    "dropdowns": {
        "noRecordsTemplate": "Keine Aufzeichnungen gefunden"
    },
    "datepicker": {
        "placeholder": "Wählen Sie ein Datum",
        "today": "heute"
    }
    }
    });
@Component({
imports: [

        TreeGridModule,
        ButtonModule
    ],
providers: [PageService,
            SortService,
            FilterService],
standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid [dataSource]='data' locale='de-DE' #treegrid
height='220' [allowPaging]='true' [allowExcelExport]='true'
[pageSettings]='pager' [treeColumnIndex]='1' [allowFiltering]='true'
[filterSettings]='filters' childMapping='subtasks'
[toolbar]='toolbarOptions'>
        <e-columns>
            <e-column field='taskID' headerText='Task ID'
textAlign='Right' width=90></e-column>
            <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
            <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
            <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
        </e-columns>
    </ejs-treegrid>`
})
export class AppComponent implements OnInit {
    public data?: Object[];
    public pager?: Object;
    public toolbarOptions?: ToolbarItems[];
    public filters?: Object;
    ngOnInit(): void {
        this.data = sampleData;
    }
}

```

```

        this.paginator = {pageSize: 7};
        this.toolbarOptions = ['Print'];
        this.filters = { type: 'Menu' };
    }
}

```

MAIN.TS

```

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

```

Internationalization

The [Internationalization](#) library is used to globalize number, date, and time values in treegrid component using format strings in the [columns.format](#).

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { loadCldr, setCulture, setCurrencyCode } from '@syncfusion/ej2-
base';
import { formatData } from './datasource';
import { L10n } from '@syncfusion/ej2-base';
import { Component, OnInit, ViewChild } from '@angular/core';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';

L10n.load({
    'de-DE': {
        'treegrid': {
            'EmptyRecord': 'Keine Aufzeichnungen angezeigt',
            'Expand All': 'Alle erweitern',
            'Collapse All': 'Alles einklappen',
            'Print': 'Drucken',
            'Pdfexport': 'PDF-Export',
            'Excelexport': 'Excel-Export',
            'Wordexport': 'Word-Export',
            'FilterButton': 'Filter',
            'ClearButton': 'klar',
            'StartsWith': 'Beginnt mit',
            'EndsWith': 'Endet mit',
            'Contains': 'Enthält',
            'Equal': 'Gleich',
            'NotEqual': 'Nicht gleich',
            'LessThan': 'Weniger als',
            'LessThanOrEqual': 'Weniger als oder gleich',
            'GreaterThan': 'Größer als',
            'GreaterThanOrEqual': 'Größer als oder gleich',
            'EnterValue': 'Geben Sie den Wert ein',
            'FilterMenu': 'Filter'
        }
    },

```

```

        'pager': {
            'currentPageInfo': '{0} von {1} Seiten',
            'totalItemsInfo': '({0} Beiträge)',
            'firstPageTooltip': 'Zur ersten Seite',
            'lastPageTooltip': 'Zur letzten Seite',
            'nextPageTooltip': 'Zur nächsten Seite',
            'previousPageTooltip': 'Zurück zur letzten Seit',
            'nextPagerTooltip': 'Zum nächsten Pager',
            'previousPagerTooltip': 'Zum vorherigen Pager'
        },
        "dropdowns": {
            "noRecordsTemplate": "Keine Aufzeichnungen gefunden"
        },
        "datepicker": {
            "placeholder": "Wählen Sie ein Datum",
            "today": "heute"
        }
    }
});
@Component({
imports: [

    TreeGridModule,
    ButtonModule

],
providers: [PageService,
            SortService,
            FilterService],
standalone: true,
    selector: 'app-container',
    template: `<ejs-treegrid [dataSource]='data' locale='de-DE' #treegrid
height='220' [allowPaging]='true' [allowExcelExport]='true'
[pageSettings]='pager' [treeColumnIndex]='1' [allowFiltering]='true'
[filterSettings]='filters' childMapping='subtasks'
[toolbar]='toolbarOptions'>
    <e-columns>
        <e-column field='orderId' headerText='Order ID'
textAlign='Right' width=90></e-column>
        <e-column field='orderName' headerText='Order Name'
textAlign='Left' width=180></e-column>
        <e-column field='price' headerText='Price' textAlign='Right'
type='number' [format]='formats' width=120></e-column>
    </e-columns>
</ejs-treegrid>`
})
export class AppComponent implements OnInit {
    public data?: Object[];
    public pager?: Object;
    public toolbarOptions?: ToolbarItems[];
    public filters?: Object;
    public formats?: Object;
    ngOnInit(): void {
        setCulture('de');
        setCurrencyCode('EUR');
        this.data = formatData;
        this.pager = {pageSize: 7};
        this.toolbarOptions = ['Print'];
    }
}

```

```

    this.filters = { type: 'Menu' };
    loadCldr('./currencies.json',
    './numbers.json',
    './ca-gregorian.json',
    './timeZoneNames.json',
    './numberingSystems.json');
    this.formats = {
        format: 'C2', useGrouping: false,
        minimumSignificantDigits: 1, maximumSignificantDigits: 3,
    currency: 'EUR'
    };
  }
}

```

MAIN.TS

```

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

```

* In the above sample, Price column is formatted by NumberFormatOptions.

* By default, [locale](#) value is en-US. If you want to change the en-US culture to a different culture, you have to change the [locale](#) accordingly.

Right to left (RTL)

RTL provides an option to switch the text direction and layout of the TreeGrid component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL Grid, set the [enableRtl](#) to true.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { L10n } from '@syncfusion/ej2-base';
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';
import { ToolbarItems } from '@syncfusion/ej2-treegrid';
import { Filter } from '@syncfusion/ej2-angular-treegrid';
L10n.load({
  'ar-AE': {
    'treegrid': {
      'EmptyRecord': "لا سجلات لعرضها",
      'Print': "طباعة",
      'FilterButton': "منقي",
      'ClearButton': "واضح",
      'StartsWith': "ابدا ب",
      'EndsWith': "ينتهي مع",
      'Contains': "يحتوي على",
      'Equal': "مساو",

```

```

        "NotEqual": "غير متساوي",
        "LessThan": "أقل من",
        "LessThanOrEqual": "اصغر من أو يساوي",
        "GreaterThan": "أكثر من",
        "GreaterThanOrEqual": "أكبر من أو يساوي",
        "ChooseDate": "اختر تاريخا",
        "EnterValue": "أدخل القيمة",
        "FilterMenu": "منقي"
    },
    'pager': {
        'currentPageInfo': '{0} صفحة 1 {من}',
        'totalItemsInfo': '({0} العناصر)',
        'firstPageTooltip': 'انتقل إلى الصفحة الأولى',
        'lastPageTooltip': 'انتقل إلى الصفحة الأخيرة',
        'nextPageTooltip': 'انتقل إلى الصفحة التالية',
        'previousPageTooltip': 'انتقل إلى الصفحة السابقة',
        'nextPagerTooltip': 'الذهاب إلى بيجر المقبل',
        'previousPagerTooltip': 'الذهاب إلى بيجر السابقة'
    },
    "dropdowns": {
        "noRecordsTemplate": "لا توجد سجلات"
    },
    "datepicker": {
        "placeholder": "اختر تاريخا",
        "today": "اليوم"
    }
    }
    });
@Component({
imports: [

        TreeGridModule,
        ButtonModule
    ],
providers: [PageService,
            SortService,
            FilterService],
standalone: true,
selector: 'app-container',
template: `<ejs-treegrid [dataSource]='data' [enableRtl]='true'
locale='ar-AE' #treegrid height='220' [allowPaging]='true'
[allowExcelExport]='true' [pageSettings]='pager' [treeColumnIndex]='1'
[allowFiltering]='true' [filterSettings]='filters' childMapping='subtasks'
[toolbar]='toolbarOptions'>
    <e-columns>
        <e-column field='taskID' headerText='Task ID' textAlign='Right'
width=90></e-column>
        <e-column field='taskName' headerText='Task Name'
textAlign='Left' width=180></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' format='yMd' width=120></e-column>
        <e-column field='duration' headerText='Duration'
textAlign='Right' width=110></e-column>
    </e-columns>
</ejs-treegrid>`
    })
export class AppComponent implements OnInit {

```

```

public data?: Object[];
public pager?: Object;
public toolbarOptions?: ToolbarItems[];
public filters?: Filter | any;
ngOnInit(): void {
    this.data = sampleData;
    this.pager = {pageSize: 7};
    this.toolbarOptions = ['Print'];
    this.filters = { type: 'Menu' };
}
}

```

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

- [Internationalization](#)
- [Localization](#)

You can refer to our [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Treegrid styling in Angular Treegrid component

To modify the TreeGrid appearance, you need to override the default CSS of treegrid. Please find the list of CSS classes and its corresponding section in treegrid. Also, you have an option to create your own custom theme for all the Angular controls using our [Theme Studio](#).

Section | CSS class | Purpose of CSS class

Root | e-treegrid | These classes are in this root element (div) of the treegrid control.

Header | e-gridheader | This class is added in the root element of header element. In this class, You can override thin line between header and content of the treegrid.

| e-table | This class is added at 'table' of the treegrid header. This CSS class makes table width as 100 %.

| e-columnheader | This class is added at 'tr' of the treegrid header.

| e-headercell | This class is added in 'th' element of treegrid header. You can override background color of header and border color.

| e-headercelldiv | This class is added in div which presents 'th' element in the header. We recommend you to use the e-headercelldiv to override skeleton of header.

Body | e-gridcontent | This class is added at root of body content. This is to override background color of the body.

| e-table | This class is added to table of content. This CSS class makes table width as 100 %.

| |e-altrow| This class is added to alternate rows of treegrid. This is to override alternate row color of the treegrid.

| |e-rowcell| This class is added to all cells in the treegrid. This is to override cells appearance and styling.

| |e-groupcaption| This class is added to the 'td' of group caption which is to change the background color of caption cell.

| |e-selectionbackground| This class is added to rowcell's of the treegrid. This is override selection.

| |e-hover| This class adds to row of treegrid, while hovering the treegrid rows.

Pager |e-pager| This class is added to root element of the pager. This to change appearance of the background color and color of font.

| |e-pagercontainer| This class is added to numeric items of the pager.

| |e-parentmsgbar| This class is added to pager info of the pager.

Summary |e-gridfooter| This class is added to root of the summary div.

| |e-summaryrow| This class is added to rows of treegrid summary.

| |e-summarycell| This class is added to cells of summary row. This to override background color of summary.

The style customization works only when we elevate the CSS to global scope using the encapsulation: ViewEncapsulation.None

If you need to apply style for ViewEncapsulation other than None, use ng-deep like shown in the below example code snippet,

```
`css
::ng-deep .e-treegrid .e-altrow {
background-color: #fafafa;
}
`
```

You can refer to our [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

How To

Refresh the data source in Angular Treegrid component

How to refresh the datasource

You can add/delete the datasource records through an external button. To reflect the datasource changes in Tree Grid, you need to assign the modified data to dataSource property.

Please follow the below steps to refresh the Tree Grid after datasource change.

Step 1:

Add/delete the datasource record by using the following code.

```
`typescript
```

```
const dataSource: object = extendArray((this.treegridObj as TreeGridComponent).dataSource as object[]);
```

```
// Added New Record.
```

```
(dataSource as object[]).unshift({ TaskID: 99, TaskName: "New Data", StartDate: new Date('02/03/2017'), Duration: 10 });
```

```
// Delete record.
```

```
(dataSource as object[]).splice(selectedRowIndex, 1);
```

```
,
```

Step 2:

Refresh the Tree Grid after the datasource change by assign the modified data to dataSource property.

```
`typescript
```

```
(this.treegridObj as TreeGridComponent).dataSource = dataSource; // Refresh the TreeGrid.
```

```
,
```

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent, extendArray } from '@syncfusion/ej2-angular-treegrid';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<button #btn1 ej2-button cssClass="e-flat"
(click)="add()">Add</button>
<button #btn2 ej2-button cssClass="e-flat"
(click)="delete()">Delete</button>
<ejs-treegrid #treegridObj [dataSource]='data' idMapping='TaskID'
parentIdMapping='parentID' [treeColumnIndex]='1' [height]='280'>
  <e-columns>
    <e-column field='TaskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
```

```

        <e-column field='TaskName' headerText='Task Name'
width='100'></e-column>
        <e-column field='StartDate' headerText='Start Date' width='90'
format="yMd" textAlign='Right'></e-column>
        <e-column field='EndDate' headerText='End Date' width='90'
format="yMd" textAlign='Right'></e-column>
        <e-column field='Duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
        <e-column field='Priority' headerText='Priority' width='90'></e-
column>
    </e-columns>
</ejs-treegrid>`,
))
export class AppComponent implements OnInit {
    public data?: Object[];
    @ViewChild('treegridObj')
    public treegridObj?: TreeGridComponent;
    ngOnInit(): void {
        this.data = projectData;
    }
    add() {
        const dataSource = extendArray((this.treegridObj as
TreeGridComponent).dataSource as object[]);
        (dataSource as object[]).unshift({ TaskID: 99, TaskName: "New Data",
StartDate: new Date('02/03/2017'), EndDate: new Date('04/04/2017'),
Duration: 10, Priority: "High" }); // Add record.
        (this.treegridObj as TreeGridComponent).dataSource = dataSource; //
Refresh the TreeGrid.
    }
    delete() {
        const selectedRow = (this.treegridObj as
TreeGridComponent).getSelectedRowIndexes().length;
        const selectedIndex = (this.treegridObj as
TreeGridComponent).getSelectedRowIndexes()[0];
        const dataSource = extendArray((this.treegridObj as
TreeGridComponent).dataSource as object[]);
        if (selectedRow > 0) {
            (dataSource as object[]).splice(selectedRowIndex, 1); // Delete
record.
        }
        else {
            alert("No records selected for delete operation");
        }
        (this.treegridObj as TreeGridComponent).dataSource = dataSource; //
Refresh the TreeGrid.
    }
}

```

MAIN.TS

```

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

```

Refresh the header by using the [refreshHeader](#) method in the tree grid.

Refresh both the header and the content by using the [refresh](#) method in the tree grid.

Refer to Syncfusion [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore Syncfusion [Angular Tree Grid example](#) to know how to present and manipulate data.

You can refer to our [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Enable/disable treegrid and its actions in Angular Treegrid component

You can enable/disable the Tree Grid and its actions by applying/removing corresponding CSS styles.

To enable/disable the Tree Grid and its actions, follow the given steps:

Step 1:

Create CSS class with custom style to override the default style of Tree Grid.

```
`css
.disabletreegrid {
  pointer-events: none;
  opacity: 0.4;
}
.wrapper {
  cursor: not-allowed;
}
`
```

Step 2:

Add/Remove the CSS class to the Tree Grid in the click event handler of Button.

```
`typescript
public click(): void {
  if (this.treegridObj && this.treegridObj.element.classList.contains('disabletreegrid')) {
    this.treegridObj.element.classList.remove('disabletreegrid');
    (document.getElementById("TreeGridParent") as HTMLElement).classList.remove('wrapper');
  }
  else if (this.treegridObj) {
    this.treegridObj.element.classList.add('disabletreegrid');
    (document.getElementById("TreeGridParent") as HTMLElement).classList.add('wrapper');
  }
}
```

In the below demo, the button click will enable/disable the Tree Grid and its actions.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit, ViewChild, ViewEncapsulation } from
'@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent, EditSettingsModel, ToolbarItems, ToolbarService,
EditService } from '@syncfusion/ej2-angular-treegrid';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  styleUrls: ['./app.disable.css'],
  encapsulation: ViewEncapsulation.None,
  providers: [ToolbarService, EditService],
  template: `<button ej2-button (click)="click()" cssClass="e-flat e-
primary" iconCss="e-icons e-play-icon" [isToggle]="true">Enable/Disable
Grid</button>
  <div id="TreeGridParent">
    <ejs-treegrid #treegridObj [dataSource]='data' idMapping='TaskID'
parentIdMapping='parentID' [treeColumnIndex]='1' [height]='210'
[editSettings]='editSettings' [toolbar]='toolbar'>
      <e-columns>
        <e-column field='TaskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
        <e-column field='TaskName' headerText='Task Name'
width='100'></e-column>
        <e-column field='StartDate' headerText='Start Date'
textAlign='Right' [format]='formatOptions' editType='datepickeredit'
[edit]='editOptions' width='90'></e-column>
        <e-column field='EndDate' headerText='End Date' width='90'
[format]='formatOptions' editType='datepickeredit' [edit]='editOptions'
textAlign='Right'></e-column>
        <e-column field='Duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
        <e-column field='Priority' headerText='Priority' width='90'></e-
column>
      </e-columns>
    </ejs-treegrid>
  </div>`,
```

```

    })
    export class AppComponent implements OnInit {
        public data: Object[] = [];
        public editOptions?: Object;
        public formatOptions?: Object;
        public editSettings?: EditSettingsModel;
        public toolbar?: ToolbarItems[];
        @ViewChild('treegridObj')
        public treegridObj?: TreeGridComponent;
        ngOnInit(): void {
            this.data = projectData;
            this.editOptions = { params: { format: 'y/M/d' } };
            this.formatOptions = { format: 'y/M/d', type: 'date' };
            this.editSettings = { allowAdding: true, allowEditing: true,
allowDeleting: true };
            this.toolbar = ['Add', 'Edit', 'Delete', 'Update', 'Cancel'];
        }
        click() {
            if (this.treegridObj &&
this.treegridObj.element.classList.contains('disabletreegrid')) {
                this.treegridObj.element.classList.remove('disabletreegrid');
                (document.getElementById("TreeGridParent") as
HTMLElement).classList.remove('wrapper');
            }
            else if (this.treegridObj) {
                this.treegridObj.element.classList.add('disabletreegrid'); (document.getEleme
ntById("TreeGridParent") as HTMLElement).classList.add('wrapper');
            }
        }
    }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Change header text dynamically in Angular Treegrid component

You can change the column [headerText](#) dynamically through an external button.

Follow the given steps to change the header text dynamically:

Step 1:

Get the column object corresponding to the field name by using the [getColumnByField](#) method.

Then change the header Text value.

```
`typescript
```

```
/ get the JSON object of the column corresponding to the field name */
```

```
const column = this.treegridObj.getColumnByField("Duration");
/ assign a new header text to the column */
column.headerText = "Changed Text";
`
```

Step 2:

To reflect the changes in the Tree Grid header, invoke the [refreshColumns](#) method.

```
`typescript
this.treegridObj.refreshColumns();
`
```

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData } from './datasource';
import { Column, TreeGridComponent } from '@syncfusion/ej2-angular-
treegrid';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<button #btn1 ej2-button cssClass="e-flat" [isToggle]="true"
(click)="click()">Change Header Text</button>
<ej2-treegrid #treegridObj [dataSource]='data' idMapping='TaskID'
parentIdMapping='parentID' [treeColumnIndex]='1' [height]='210'>
  <e-columns>
    <e-column field='TaskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
    <e-column field='TaskName' headerText='Task Name' width='100'
></e-column>
    <e-column field='StartDate' headerText='Start Date' width='90'
format='yMd' textAlign='Right'></e-column>
    <e-column field='EndDate' headerText='End Date' width='90'
format='yMd' textAlign='Right'></e-column>
    <e-column field='Duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
```

```

        <e-column field='Priority' headerText='Priority' width='90'></e-
column>
    </e-columns>
</ejs-treegrid>`,
}))
export class AppComponent implements OnInit {
    public data: Object[] = [];
    @ViewChild('treegridObj')
    public treegridObj?: TreeGridComponent;
    ngOnInit(): void {
        this.data = projectData;
    }
    click() {
        const column = this.treegridObj?.getColumnByField('Duration'); //
        get the JSON object of the column corresponding to the field name
        (column as Column).headerText = 'Changed Text'; // assign a new
        header text to the column
        this.treegridObj?.refreshColumns();
    }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Customize column styles in Angular Treegrid component

You can customise the appearance of header and content of the particular column using the [customAttributes](#) property.

To customize the Tree Grid column, follow the given steps:

Step 1:

Create a CSS class with custom style to override the default style for rowcell and headercell.

```

`css
.e-treegrid .e-rowcell.customcss{
background-color: #ecedee;
font-family: 'Bell MT';
color: 'red';
font-size: '20px';
}
.e-treegrid .e-headercell.customcss{
background-color: #2382c3;

```



```

color: white;
font-family: 'Bell MT';
font-size: '20px';
}
`

```

Step 2:

Add the custom CSS class to particular column by using [customAttributes](#) property.

```
`typescript
```

```

<e-column field='TaskName' headerText='Task Name' width='170'
[customAttributes]='customAttributes'></e-column>
`

```

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit, ViewChild, ViewEncapsulation } from
'@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  styleUrls: ['./app.component.css'],
  encapsulation: ViewEncapsulation.None,
  template: `<ejs-treegrid [dataSource]='data' idMapping='TaskID'
parentIdMapping='parentID' [treeColumnIndex]='1' [height]='317'>
    <e-columns>
      <e-column field='TaskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
      <e-column field='TaskName' headerText='Task Name' width='100'
[customAttributes]='customAttributes'></e-column>
      <e-column field='StartDate' headerText='Start Date' width='90'
format='yMd' textAlign='Right'></e-column>
      <e-column field='EndDate' headerText='End Date' width='90'
format='yMd' textAlign='Right'></e-column>
      <e-column field='Duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
    </e-columns>
  `
})

```

```

    </ejs-treegrid>`,
  })
  export class AppComponent implements OnInit {
    public data: Object[] = [];
    public customAttributes?: Object;
    ngOnInit(): void {
      this.data = projectData;
      this.customAttributes = {class: 'customcss'};
    }
  }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Custom tool tip for columns in Angular Treegrid component

You can achieve the custom tooltip([EJ2 Tooltip](#)) for Tree Grid by using the [queryCellInfo](#) event.

Render the Tooltip component for the Tree Grid cells by using the following code in the [queryCellInfo](#) event.

```

`typescript
public tooltip(args: QueryCellInfoEventArgs){
  const tooltip: Tooltip = new Tooltip({
    content: args.data[args.column.field].toString()
  });
  tooltip.appendTo(args.cell as HTMLElement);
}
`

```

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { QueryCellInfoEventArgs } from '@syncfusion/ej2-angular-grids';
import { Tooltip } from '@syncfusion/ej2-popups';

```

```

@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegridObj [dataSource]='data'
idMapping='TaskID' parentIdMapping='parentID' [treeColumnIndex]='1'
[height]='317' (queryCellInfo)='tooltip($event)' >
    <e-columns>
      <e-column field='TaskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
      <e-column field='TaskName' headerText='Task Name' width='100'
></e-column>
      <e-column field='StartDate' headerText='Start Date' width='90'
format="yMd" textAlign='Right'></e-column>
      <e-column field='EndDate' headerText='End Date' width='90'
format="yMd" textAlign='Right'></e-column>
      <e-column field='Duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
      <e-column field='Priority' headerText='Priority' width='90'></e-
column>
    </e-columns>
  </ejs-treegrid>`,
})
export class AppComponent implements OnInit {
  public data: Object[] = [];
  @ViewChild('treegridObj')
  public treegridObj?: TreeGridComponent;
  ngOnInit(): void {
    this.data = projectData;
  }
  tooltip(args: QueryCellInfoEventArgs | any) {
    const tooltip: Tooltip = new Tooltip({
      content: args.data[args.column.field].toString()
    }, args.cell as HTMLTableCellElement);
  }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Change orientation of header text in Angular Treegrid component

You can change the orientation of the header text by using the [customAttributes](#) property.

Ensure the following steps:

Step 1:

Create a CSS class with orientation style for Tree Grid header cell.

```
`css
.orientationcss .e-headercelldiv {
transform: rotate(90deg);
}
`
```

Step 2:

Add the custom CSS class to particular column by using [customAttributes](#) property.

```
`typescript
<e-column field='EndDate' headerText='End Date' width='90' format='yMd' textAlign='Right'
[customAttributes]='customAttributes' ></e-column>
`
```

Step 3:

Resize the header cell height in [create](#) event by using the following code.

```
`typescript
public setHeaderHeight() {
/ Obtain the width of the headerText content */
const textWidth: number = (document.querySelector(".orientationcss > div") as
HTMLElement).scrollWidth;
const headerCell: NodeList = document.querySelectorAll(".e-headercell");
for(let i: number = 0; i < headerCell.length; i++) {
/ Assign the obtained textWidth as the height of the headerCell */
((headerCell as any).item(i)).style.height = textWidth + 'px';
}
}
`
```

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
```

```

import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit, ViewChild, ViewEncapsulation } from
 '@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  styleUrls: ['./app.component.css'],
  encapsulation: ViewEncapsulation.None,
  template: `<ejs-treegrid #treegrid [dataSource]='data'
idMapping='TaskID' parentIdMapping='parentID' [height]='194'
[treeColumnIndex]='1' (created)='setHeaderHeight($event)' >
    <e-columns>
      <e-column field='TaskID' headerText='Task ID'
width='70' textAlign='Right'></e-column>
      <e-column field='TaskName' headerText='Task Name'
width='100' ></e-column>
      <e-column field='StartDate' headerText='Start Date'
width='90' format='yMd' textAlign='Right' ></e-column>
      <e-column field='EndDate' headerText='End Date'
width='90' format='yMd' textAlign='Center'
[customAttributes]='customAttributes' ></e-column>
      <e-column field='Duration' headerText='Duration'
width='90' textAlign='Right' ></e-column>
      <e-column field='Progress' headerText='Progress'
width='90' textAlign='Right' ></e-column>
    </e-columns>
  </ejs-treegrid>`,
})
export class AppComponent implements OnInit {
  public data: Object[] = [];
  public customAttributes?: Object;
  ngOnInit(): void {
    this.data = projectData;
    this.customAttributes = { class: 'orientationcss' };
  }
  setHeaderHeight(args: any) {
    const textWidth: number = (document.querySelector('.orientationcss >
div') as Element).scrollWidth as number; // Obtain the width of the
headerText content.
    const headerCell: NodeList = document.querySelectorAll('.e-
headercell');
    for (let i = 0; i < headerCell.length; i++) {
      // Assign the obtained textWidth as the height of the
headerCell.
      (headerCell.item(i) as HTMLElement).style.height = textWidth +
'px';
    }
  }
}

```

```
}
}
```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Render other component in column in Angular Treegrid component

You can render any components in a Tree Grid column using the template property.

Initialize the column template for your custom component. The template property renders the custom component.

In the following sample, the DropDownList is rendered in the **Priority** column.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegridObj [dataSource]='data'
idMapping='TaskID' parentIdMapping='parentID' [treeColumnIndex]='1'
[height]='315' >
    <e-columns>
      <e-column field='TaskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
      <e-column field='TaskName' headerText='Task Name' width='100'
></e-column>
      <e-column field='StartDate' headerText='Start Date'
textAlign='Right' [format]='formatOptions' width='90'></e-column>
```

```

        <e-column field='EndDate' headerText='End Date'
textAlign='Right' [format]='formatOptions' width='90'></e-column>
        <e-column field='Duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
        <e-column headerText='Priority' width='90'>
            <ng-template #template let-data>
                <div>
                    <ejs-dropdownlist value='Normal' [dataSource]='dropData'
(change)='onChange($event)' >
                        </ejs-dropdownlist>
                    </div>
                </ng-template>
            </e-column>
        </e-columns>
    </ejs-treegrid>`,
    })
    export class AppComponent implements OnInit {
        public data: Object[] = [];
        public formatOptions?: Object;
        public dropData?: string[];
        ngOnInit(): void {
            this.data = projectData;
            this.formatOptions = { format: 'y/M/d', type: 'date' };
            this.dropData = ['Normal', 'Low', 'High', 'Critical', 'Breaker'];
        }
        public onChange(args: any): void {
            /** Event will trigger when you have change the value in dropdown
column */
            alert(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));

```

You can refer to our [Angular Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Customize the icon for column menu in Angular Treegrid component

You can customize the column menu icon by overriding the default Tree Grid class **.e-icons.e-columnmenu** with a custom property **content** as mentioned below,

```

`css
.e-treegrid .e-columnheader .e-icons.e-columnmenu::before {
    content: "\e903";
}
`

```

In the below sample, Tree Grid is rendered with a customized column menu icon.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, OnInit, ViewChild, ViewEncapsulation } from
'@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent, ColumnMenuService } from '@syncfusion/ej2-
angular-treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  styleUrls: ['./app.component.css'],
  encapsulation: ViewEncapsulation.None,
  providers: [ColumnMenuService],
  template: `<ejs-treegrid #treegrid [dataSource]='data'
idMapping='TaskID' parentIdMapping='parentID' [treeColumnIndex]='1'
[height]='315' showColumnMenu=true>
    <e-columns>
      <e-column field='TaskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
      <e-column field='TaskName' headerText='Task Name' width='100'
></e-column>
      <e-column field='StartDate' headerText='Start Date' width='90'
format='yMd' textAlign='Right' ></e-column>
      <e-column field='EndDate' headerText='End Date' width='90'
format='yMd' textAlign='Right'></e-column>
      <e-column field='Duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
      <e-column field='Priority' headerText='Priority' width='90'></e-
column>
    </e-columns>
  </ejs-treegrid>`,
})
export class AppComponent implements OnInit {
  public data: Object[] = [];
  ngOnInit(): void {
    this.data = projectData;
  }
}
```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Customize the edit dialog in Angular Treegrid component

You can customize the appearance of the edit dialog in the [actionComplete](#) event based on **requestType** as **beginEdit** or **add**.

In the below example, we have changed the dialog's header text for editing and adding records.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent, EditSettingsModel, ToolbarItems, EditService,
ToolbarService } from '@syncfusion/ej2-angular-treegrid';
import { DialogEditEventArgs } from '@syncfusion/ej2-angular-grids';
import { Dialog } from '@syncfusion/ej2-popups';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  providers: [EditService, ToolbarService],
  template: `<ejs-treegrid #treegrid [dataSource]='data'
idMapping='TaskID' parentIdMapping='parentID' [treeColumnIndex]='1'
[height]='265' [editSettings]='editSettings' [toolbar]='toolbar'
(actionComplete)="actionComplete($event)" >
    <e-columns>
      <e-column field='TaskID' headerText='Task ID'
isPrimaryKey='true' width='70' textAlign='Right'></e-column>
      <e-column field='TaskName' headerText='Task Name' width='100'
></e-column>
      <e-column field='StartDate' headerText='Start Date'
textAlign='Right' [format]='formatOptions' editType='datepickeredit'
[edit]='editOptions' width='90'></e-column>
```

```

        <e-column field='EndDate' headerText='End Date' width='90'
[format]='formatOptions' editType='datepickeredit' [edit]='editOptions'
textAlign='Right'></e-column>
        <e-column field='Duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
        <e-column field='Priority' headerText='Priority' width='90'></e-
column>
    </e-columns>
</ejs-treegrid>`,
}))
export class AppComponent implements OnInit {
    public data: Object[] = [];
    public editSettings?: EditSettingsModel;
    public toolbar?: ToolbarItems[];
    public editOptions?: Object;
    public formatOptions?: Object;
    ngOnInit(): void {
        this.data = projectData;
        this.editOptions = { params: { format: 'y/M/d' } };
        this.formatOptions = { format: 'y/M/d', type: 'date' };
        this.editSettings = { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Dialog' };
        this.toolbar = ['Add', 'Edit', 'Delete', 'Update', 'Cancel'];
    }
    actionComplete(args: DialogEditEventArgs) {
        if ((args.requestType === 'beginEdit' || args.requestType ===
'add')) {
            const dialog = args.dialog as Dialog;
            const TaskName = 'TaskName';
            dialog.height = 400;
            // change the header of the dialog
            dialog.header = args.requestType === 'beginEdit' ? 'Record of '
+ (args as any).rowData[TaskName] : 'New Customer';
        }
    }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Cascading drop down list with treegrid editing in Angular Treegrid component

You can achieve the Cascading DropDownList with Tree Grid Editing by using the Cell Edit Template feature.

In the below demo, Cascading DropDownList rendered for **Priority** and **Duration** column.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'

```

```

import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent, EditSettingsModel, ToolbarItems, EditService,
ToolbarService } from '@syncfusion/ej2-angular-treegrid';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { DataManager, Query } from '@syncfusion/ej2-data';
import { IEditCell } from '@syncfusion/ej2-angular-grids';
@Component({
imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
],
providers: [PageService,
            SortService,
            FilterService],
standalone: true,
    selector: 'app-container',
    providers: [EditService, ToolbarService],
    template: `<ejs-treegrid #treegridObj [dataSource]='data'
idMapping='TaskID' parentIdMapping='parentID' [treeColumnIndex]='1'
[editSettings]='editSettings' [toolbar]='toolbar' [height]='273'>
    <e-columns>
        <e-column field='TaskID' headerText='Task ID'
isPrimaryKey='true' width='70' textAlign='Right'></e-column>
        <e-column field='TaskName' headerText='Task Name' width='100'
></e-column>
        <e-column field='StartDate' headerText='Start Date' width='100'
format='yMd' textAlign='Right' editType='datepickeredit'></e-column>
        <e-column field='EndDate' headerText='End Date' width='90'
format='yMd' textAlign='Right' editType='datepickeredit'></e-column>
        <e-column field='Priority' headerText='Priority' width='90'
editType='dropdownedit' [edit]='priorityParams'></e-column>
        <e-column field='Duration' headerText='Duration' width='90'
textAlign='Right' editType='dropdownedit' [edit]='durationParams'></e-
column>
        <e-column field='Progress' headerText='Progress' width='90'
textAlign='Right'></e-column>
    </e-columns>
</ejs-treegrid>`,
})
export class AppComponent implements OnInit {
    public data: Object[] = [];
    public editSettings?: EditSettingsModel;
    public toolbar?: ToolbarItems[];
    public priorityParams?: IEditCell;
    public durationParams?: IEditCell;
    public priorityElem?: HTMLElement;
    public priorityObj?: DropDownList;
    public durationElem?: HTMLElement;

```

```

public durationObj?: DropDownList;
@ViewChild('treegridObj')
public treegridObj?: TreeGridComponent;
public priorityData: { [key: string]: Object }[] = [
  { priorityName: 'Normal', priorityId: '1' },
  { priorityName: 'High', priorityId: '2' }
];
public durationData : { [key: string]: Object }[] = [
  { durationValue: 2, priorityId: '1', durationId: 2 },
  { durationValue: 3, priorityId: '1', durationId: 3 },
  { durationValue: 4, priorityId: '1', durationId: 4 },
  { durationValue: 11, priorityId: '2', durationId: 11 },
  { durationValue: 15, priorityId: '2', durationId: 15 },
  { durationValue: 20, priorityId: '2', durationId: 20 }
];
ngOnInit(): void {
  this.data = projectData;
  this.editSettings = { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Row' };
  this.toolbar = ['Add', 'Edit', 'Delete', 'Update', 'Cancel'];
  this.priorityParams = {
    create: ()=>{
      this.priorityElem = document.createElement('input');
      return this.priorityElem;
    },
    read: ()=>{
      return this.priorityObj?.text;
    },
    destroy: ()=>{
      this.priorityObj?.destroy();
    },
    write: ()=>{
      this.priorityObj = new DropDownList({
        dataSource: new DataManager(this.priorityData),
        fields: { value: 'priorityId', text: 'priorityName' },
        change: () => {
          (this.durationObj as DropDownList).enabled = true;
          let tempQuery: Query = new Query().where('priorityId',
'equal', this.priorityObj?.value);
          (this.durationObj as DropDownList).query = tempQuery;
          (this.durationObj as any).text = undefined;
          this.durationObj?.dataBind();
        },
        placeholder: 'Select a priority',
        floatLabelType: 'Never'
      });
      this.priorityObj.appendTo(this.priorityElem);
    }
  };
  this.durationParams = {
    create: ()=>{
      this.durationElem = document.createElement('input');
      return this.durationElem;
    },
    read: ()=>{
      return this.durationObj?.text;
    },
    destroy: ()=>{

```

```

        this.durationObj?.destroy();
    },
    write: () => {
        this.durationObj = new DropDownList({
            dataSource: new DataManager(this.durationData),
            fields: { value: 'durationId', text: 'durationValue' },
            enabled: false,
            placeholder: 'Select a duration',
            floatLabelType: 'Never'
        });
        this.durationObj.appendTo(this.durationElem);
    }
}
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Provide custom data source and enabling filtering to drop down list in Angular Treegrid component

You can provide data source to the DropDownList by using the **params** of [columns.edit](#) property.

While setting new data source using edit params, you must specify a new **query** property for the DropDownList as follows,

```

`typescript
public priorityParams : IEditCell = {
  params: {
    actionComplete: () => false,
    allowFiltering: true,
    dataSource: new DataManager(this.priorityData),
    fields: { text: "countryName", value: "countryName"},
    query: new Query()
  }
};
`

```

You can also enable filtering for the DropDownList by passing the [allowFiltering](#) as **true** to the edit params.

In the below demo, DropDownList is rendered with custom [dataSource](#) for the *Priority* column and enabled filtering to search DropDownList items.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent, EditService, ToolbarService, EditSettingsModel,
ToolbarItems } from '@syncfusion/ej2-angular-treegrid';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { Query, DataManager } from '@syncfusion/ej2-data';
import { IEditCell } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  providers: [EditService, ToolbarService],
  template: `<ejs-treegrid #treegrid [dataSource]='data'
idMapping='TaskID' parentIdMapping='parentID' [treeColumnIndex]='1'
[height]='265' [editSettings]='editSettings'
(queryCellInfo)='tooltip($event)' >
    <e-columns>
      <e-column field='TaskID' headerText='Task ID'
isPrimaryKey='true' width='70' textAlign='Right'></e-column>
      <e-column field='TaskName' headerText='Task Name' width='100'
></e-column>
      <e-column field='StartDate' headerText='Start Date' width='100'
[format]='formatOptions' editType='datepickeredit' textAlign='Right'></e-
column>
      <e-column field='EndDate' headerText='End Date' width='100'
[format]='formatOptions' editType='datepickeredit' textAlign='Right'></e-
column>
      <e-column field='Duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
      <e-column field='Priority' headerText='Priority' width='90'
editType='dropdownedit'
[edit]='priorityParams'></e-column>
    </e-columns>
  </ejs-treegrid>`,
})
export class AppComponent implements OnInit {
  tooltip($event: any) {
```

```

throw new Error('Method not implemented.');
```

```

    public data: Object[] = [];
    public editSettings?: EditSettingsModel;
    public toolbar?: ToolbarItems[];
    public editOptions?: Object;
    public formatOptions?: Object;
    public priorityParams?: IEditCell;
    public priorityData : object[] = [
        { priorityName: 'Normal', priorityId: '1' },
        { priorityName: 'High', priorityId: '2' },
        { priorityName: 'Low', priorityId: '3' },
        { priorityName: 'Critical', priorityId: '4' },
        { priorityName: 'Breaker', priorityId: '5' }
    ];

    ngOnInit(): void {
        this.data = projectData;
        this.editSettings = { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Row' };
        this.toolbar = ['Add', 'Edit', 'Delete'];
        this.editOptions = { params: { format: 'y/M/d' } };
        this.formatOptions = { format: 'y/M/d', type: 'date' };
        this.priorityParams = {
            params: {
                actionComplete: () => false,
                allowFiltering: true,
                dataSource: new DataManager(this.priorityData),
                fields: { text: 'priorityName', value: 'priorityName' },
                query: new Query()
            }
        };
    }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Restrict decimal points while treegrid editing in Angular Treegrid component

By default, the number of decimal places will be restricted to two in the `NumericTextBox` while editing the numeric column. We can restrict to type the decimal points in a `NumericTextBox` by using the **`validateDecimalOnType`** and **`decimals`** properties of `NumericTextBox`.

In the below demo, while editing the row we have restricted to type the decimal point value in the NumericTextBox of **Price** column.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
```

```

import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { stackedData } from './datasource';
import { TreeGridComponent, EditSettingsModel, ToolbarItems, ToolbarService,
EditService } from '@syncfusion/ej2-angular-treegrid';
import { IEditCell } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  providers: [ToolbarService, EditService],
  template: `<ejs-treegrid #treegridObj [dataSource]='data'
childMapping='subtasks' [treeColumnIndex]='1' [height]='265'
[editSettings]='editSettings' [toolbar]='toolbar'>
    <e-columns>
      <e-column field='orderID' headerText='Order ID' width='70'
textAlign='Right' isPrimaryKey='true'></e-column>
      <e-column field='orderName' headerText='Order Name' width='100'
></e-column>
      <e-column field='orderDate' headerText='Order Date'
textAlign='Right' [format]='formatOptions' editType= 'datepickeredit'
width='100'></e-column>
      <e-column field='shippedDate' headerText='Shipped Date'
textAlign='Right' [format]='formatOptions' editType= 'datepickeredit'
width='100'></e-column>
      <e-column field='shipMentCategory' headerText='Shipment
Category' width='100' ></e-column>
      <e-column field='units' headerText='Units' width='90'
textAlign='Right' editType= 'numericedit'></e-column>
      <e-column field='price' headerText='Price' width='90'
textAlign='Right' [format]='numericFormatOptions' editType= 'numericedit'
[edit]='numericParams' ></e-column>
    </e-columns>
  </ejs-treegrid>`,
})
export class AppComponent implements OnInit {
  public data: Object[] = [];
  public formatOptions?: Object;
  public numericFormatOptions?: Object;
  public editSettings?: EditSettingsModel;
  public toolbar?: ToolbarItems[];
  public numericParams?: IEditCell
  ngOnInit(): void {
    this.data = stackedData;
  }
}

```



```

        this.formatOptions = { format: 'y/M/d', type: 'date' };
        this.numericFormatOptions = {format: 'c2'}
        this.editSettings = { allowEditing: true, allowAdding: true,
allowDeleting: true };
        this.toolbar = ['Add', 'Edit', 'Delete', 'Update', 'Cancel'];
        this.numericParams = { params: {
            validateDecimalOnType: true,
            decimals: 0,
            format: 'N' }
        };
    }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Exporting filtered data in Angular Treegrid component

You can export the filtered data by defining the resulted data in [PdfExportProperties.dataSource](#) before export.

In the below Pdf exporting demo, We have gotten the filtered data from the filteredResult of Tree Grid filterModule and then defines the resulted data in [PdfExportProperties.dataSource](#) and pass it to [pdfExport](#) method.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent, ToolbarItems, ToolbarService, PdfExportService,
PageService, FilterService } from '@syncfusion/ej2-angular-treegrid';
import { ClickEventArgs } from '@syncfusion/ej2-angular-navigations';
import { PdfExportProperties } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,

```

```

        FilterService],
standalone: true,
    selector: 'app-container',
    providers: [ToolbarService, PdfExportService, PageService,
FilterService],
    template: `<ejs-treegrid #treegridObj [dataSource]='data'
idMapping='TaskID' parentIdMapping='parentID' [treeColumnIndex]='1'
[allowFiltering]='true' [allowPaging]='true' [pageSettings]='initialPage'
[allowPdfExport]='true' [toolbar]='toolbarOptions'
(toolbarClick)='toolbarClick($event)' ` >
        <e-columns>
            <e-column field='TaskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
            <e-column field='TaskName' headerText='Task Name' width='100'
></e-column>
            <e-column field='StartDate' headerText='Start Date'
textAlign='Right' [format]='formatOptions' editType='datepickeredit'
[edit]='editOptions' width='100'></e-column>
            <e-column field='EndDate' headerText='Start Date'
textAlign='Right' [format]='formatOptions' editType='datepickeredit'
[edit]='editOptions' width='100'></e-column>
            <e-column field='Duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
            <e-column field='Priority' headerText='Priority' width='90'></e-
column>
        </e-columns>
    </ejs-treegrid>`,
))
export class AppComponent implements OnInit {
    public data: Object[] = [];
    public editOptions?: Object;
    public formatOptions?: Object;
    public toolbarOptions?: ToolbarItems[];
    public initialPage?: object;
    @ViewChild('treegridObj')
    public treegridObj?: TreeGridComponent;
    ngOnInit(): void {
        this.data = projectData;
        this.editOptions = { params: { format: 'y/M/d' } };
        this.formatOptions = { format: 'y/M/d', type: 'date' };
        this.toolbarOptions = ['PdfExport'];
        this.initialPage = { pageCount: 5, pageSize: 5 };
    }
    toolbarClick(args: ClickEventArgs) {
        if (this.treegridObj && args.item.text === 'PDF Export') {
            let pdfdata;
            pdfdata = this.treegridObj.filterModule.filteredResult;
            const exportProperties = {
                dataSource: pdfdata,
            };
            if (this.treegridObj) {
                this.treegridObj.pdfExport(exportProperties);
            }
        }
    }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Exporting selected data in Angular Treegrid component

You can export the selected records data by passing it to [PdfExportProperties.dataSource](#) or [ExcelExportProperties.dataSource](#) property in the [toolbarClick](#) event.

In the below exporting demo, we can get the selected records using [getSelectedRecords](#) method and pass the selected data to [pdfExport](#) or [excelExport](#) methods using respective export properties..

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent, ToolbarItems, ToolbarService, PdfExportService,
PageService, ExcelExportService, SelectionSettingsModel } from
 '@syncfusion/ej2-angular-treegrid';
import { ClickEventArgs } from '@syncfusion/ej2-angular-navigations';
import { PdfExportProperties } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  providers: [ToolbarService, PdfExportService, PageService,
ExcelExportService],
  template: `<ejs-treegrid #treegridObj [dataSource]='data'
idMapping='TaskID' parentIdMapping='parentID' [treeColumnIndex]='1'
[allowPaging]='true' [pageSettings]='initialPage' [allowPdfExport]='true'
[allowExcelExport]='true' [toolbar]='toolbarOptions'
(toolbarClick)='toolbarClick($event)'
[selectionSettings]='selectionSettings'>
    <e-columns>
```

```

        <e-column field='TaskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
        <e-column field='TaskName' headerText='Task Name' width='100'
></e-column>
        <e-column field='StartDate' headerText='Start Date'
textAlign='Right' [format]='formatOptions' editType='datepickeredit'
[edit]='editOptions' width='100'></e-column>
        <e-column field='EndDate' headerText='End Date'
textAlign='Right' [format]='formatOptions' editType='datepickeredit'
[edit]='editOptions' width='100'></e-column>
        <e-column field='Duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
        <e-column field='Priority' headerText='Priority' width='90'></e-
column>
    </e-columns>
</ejs-treegrid>,
))
export class AppComponent implements OnInit {
    public data: Object[] = [];
    public editOptions?: Object;
    public formatOptions?: Object;
    public toolbarOptions?: ToolbarItems[];
    public selectionSettings?: SelectionSettingsModel;
    public initialPage?: object;
    @ViewChild('treegridObj')
    public treegridObj?: TreeGridComponent;
    ngOnInit(): void {
        this.data = projectData;
        this.editOptions = { params: { format: 'y/M/d' } };
        this.formatOptions = { format: 'y/M/d', type: 'date' };
        this.initialPage = { pageCount: 5, pageSize: 5 };
        this.toolbarOptions = ['PdfExport', 'ExcelExport'];
        this.selectionSettings = { type: 'Multiple' };
    }
    toolbarClick(args: ClickEventArgs) {
        if (this.treegridObj && args.item.text === 'PDF Export') {
            const selectedRecords =
this.treegridObj.getSelectedRecords();
            const exportProperties = {
                dataSource: selectedRecords,
            };
            this.treegridObj.pdfExport(exportProperties);
        }
        else if (this.treegridObj && args.item.text === 'Excel Export') {
            const selectedRecords =
this.treegridObj.getSelectedRecords();
            const exportProperties = {
                dataSource: selectedRecords,
            };
            this.treegridObj.excelExport(exportProperties);
        }
    }
}
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Show spinner while exporting in Angular Treegrid component

You can show/ hide spinner component while exporting the Tree Grid using [showSpinner/ hideSpinner](#) methods. You can use [toolbarClick](#) event to show spinner before exporting and hide a spinner in the [pdfExportComplete](#) or [excelExportComplete](#) event after the exporting.

In the [toolbarClick](#) event, based on the parameter **args.item.text** as **PDF Export** or **Excel Export** we can call the [showSpinner](#) method from Tree Grid instance.

In the [pdfExportComplete](#) or [excelExportComplete](#) event, We can call the [hideSpinner](#) method.

In the below demo, we have rendered the default spinner component when exporting the Tree Grid.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent, ToolbarItems, ToolbarService, PageService, PdfExportService, ExcelExportService } from '@syncfusion/ej2-angular-treegrid';
import { ClickEventArgs } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService, SortService, FilterService],
  standalone: true,
  selector: 'app-container',
  providers: [ToolbarService, PageService, PdfExportService, ExcelExportService],
  template: `<ejs-treegrid #treegridObj [dataSource]='data'
  idMapping='TaskID' parentIdMapping='parentID'
  [treeColumnIndex]='1' [allowPaging]='true' [pageSettings]='initialPage'
  [toolbar]='toolbarOptions'
  [allowPdfExport]='true' [allowExcelExport]='true'
  (excelExportComplete)='excelExportComplete()'`
})
```

```

(pdfExportComplete)='pdfExportComplete()'
(toolbarClick)='toolbarClick($event) '>
    <e-columns>
        <e-column field='TaskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
        <e-column field='TaskName' headerText='Task Name' width='100'
></e-column>
        <e-column field='StartDate' headerText='Start Date'
textAlign='Right' [format]='formatOptions' width='100'></e-column>
        <e-column field='EndDate' headerText='Start Date'
textAlign='Right' [format]='formatOptions' width='100'></e-column>
        <e-column field='Duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
        <e-column field='Priority' headerText='Priority' width='90'></e-
column>
    </e-columns>
</ejs-treegrid>`,
})
export class AppComponent implements OnInit {
    public data: Object[] = [];
    public formatOptions?: Object;
    public toolbarOptions?: ToolbarItems[];
    public initialPage?: object;
    @ViewChild('treegridObj')
    public treegridObj?: TreeGridComponent;
    ngOnInit(): void {
        this.data = projectData;
        this.formatOptions = { format: 'y/M/d', type: 'date' };
        this.toolbarOptions = ['PdfExport', 'ExcelExport'];
        this.initialPage = { pageCount: 5, pageSize: 5 };
    }
    toolbarClick(args: ClickEventArgs) {
        if (this.treegridObj && args.item.text === 'PDF Export') {
            this.treegridObj.showSpinner();
            this.treegridObj.pdfExport();
        }
        else if (this.treegridObj && args.item.text === 'Excel Export') {
            this.treegridObj.showSpinner();
            this.treegridObj.excelExport();
        }
    }
    pdfExportComplete() {
        if (this.treegridObj) {
            this.treegridObj.hideSpinner();
        }
    }
    excelExportComplete() {
        if (this.treegridObj) {
            this.treegridObj.hideSpinner();
        }
    }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Passing parameter to server exporting in Angular Treegrid component

You can pass the additional parameter in the [query](#) property by invoking [addParams](#) method. In the [toolbarClick](#) event, you can define params as key and value pair so it will receive at the server side when exporting.

In the below example, we have passed *recordcount* as 12 using [addParams](#) method.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent, ToolbarItems, ToolbarService, PageService,
PdfExportService, ExcelExportService } from '@syncfusion/ej2-angular-
treegrid';
import { Query } from '@syncfusion/ej2-data';
import { ClickEventArgs } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  providers: [ToolbarService, PageService,
    PdfExportService, ExcelExportService],
  template: `<ejs-treegrid #treegridObj [dataSource]='data'
idMapping='TaskID'
  parentIdMapping='parentID' [treeColumnIndex]='1' [allowPaging]='true'
[pageSettings]='initialPage'
  [toolbar]='toolbarOptions' [allowPdfExport]='true'
[allowExcelExport]='true'
  (excelExportComplete)='excelExportComplete()'
(pdfExportComplete)='pdfExportComplete()'
  (toolbarClick)='toolbarClick($event)'>
    <e-columns>
      <e-column field='TaskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
```

```

        <e-column field='TaskName' headerText='Task Name' width='100'
    ></e-column>
        <e-column field='StartDate' headerText='Start Date'
    textAlign='Right' [format]='formatOptions' width='100'></e-column>
        <e-column field='EndDate' headerText='Start Date'
    textAlign='Right' [format]='formatOptions' width='100'></e-column>
        <e-column field='Duration' headerText='Duration' width='90'
    textAlign='Right'></e-column>
        <e-column field='Priority' headerText='Priority' width='90'></e-
column>
    </e-columns>
</ejs-treegrid>,
))
export class AppComponent implements OnInit {
    public data: Object[] = [];
    public formatOptions?: Object;
    public toolbarOptions?: ToolbarItems[];
    public initialPage?: object;
    public queryClone?: any;
    @ViewChild('treegridObj')
    public treegridObj?: TreeGridComponent;
    ngOnInit(): void {
        this.data = projectData;
        this.formatOptions = { format: 'y/M/d', type: 'date' };
        this.toolbarOptions = ['PdfExport', 'ExcelExport'];
        this.initialPage = { pageCount: 5, pageSize: 5 };
    }
    toolbarClick(args: ClickEventArgs) {
        if (this.treegridObj && args.item.text === 'PDF Export') {
            this.queryClone = this.treegridObj.query;
            this.treegridObj.query = new Query().addParams("recordcount", "12");
            this.treegridObj.pdfExport();
        }
        else if (this.treegridObj && args.item.text === 'Excel Export') {
            this.queryClone = this.treegridObj.query;
            this.treegridObj.query = new Query().addParams("recordcount", "12");
            this.treegridObj.excelExport();
        }
    }
    pdfExportComplete() {
        if (this.treegridObj) {
            this.treegridObj.query = this.queryClone;
        }
    }
    excelExportComplete() {
        if (this.treegridObj) {
            this.treegridObj.query = this.queryClone;
        }
    }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Customize pager drop down in Angular Treegrid component

To customize default values of pager dropdown, you need to define [pageSizes](#) as array of strings.

APP.COMPONENT.TS

```
import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent, PageService } from '@syncfusion/ej2-angular-
treegrid';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  providers: [ PageService ],
  template: `<ejs-treegrid #treegridObj [dataSource]='data'
idMapping='TaskID' parentIdMapping='parentID' [treeColumnIndex]='1'
[height]='268' [allowPaging]='true' [pageSettings]='initialPage' >
  <e-columns>
    <e-column field='TaskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
    <e-column field='TaskName' headerText='Task Name' width='200'
></e-column>
    <e-column field='StartDate' headerText='Start Date'
textAlign='Right' [format]='formatOptions' editType= 'datepickeredit'
width='90'></e-column>
    <e-column field='EndDate' headerText='Start Date'
textAlign='Right' [format]='formatOptions' editType= 'datepickeredit'
width='90'></e-column>
    <e-column field='Duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
    <e-column field='Progress' headerText='Progress' width='80'
textAlign='Right'></e-column>
    <e-column field='Priority' headerText='Priority' width='90'></e-
column>
  </e-columns>
</ejs-treegrid>`,
```

```

}))
export class AppComponent implements OnInit {
  public data: Object[] = [];
  public formatOptions?: Object;
  public initialPage?: object;
  ngOnInit(): void {
    this.data = projectData;
    this.formatOptions = { format: 'y/M/d', type: 'date' };
    this.initialPage = { pageSizes: ['5', '10', 'All'], };
  }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Add parameter for filtering in Angular Treegrid component

You can customize the default settings of the components which are used in Menu filter by using params of filter property in column definition.

In the below sample, TaskID and Duration Columns are numeric columns, while opening the filter dialog you can see that NumericTextBox with spin button is displayed to change/set the filter value. Now using the params option we hide the spin button in NumericTextBox for TaskID Column.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent, FilterService, FilterSettingsModel } from
 '@syncfusion/ej2-angular-treegrid';
import { IFilter } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [
    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,

```

```

    selector: 'app-container',
    providers: [ FilterService ],
    template: `<ejs-treegrid #treegridObj [dataSource]='data'
idMapping='TaskID' parentIdMapping='parentID' [treeColumnIndex]='1'
[height]='273' [allowFiltering]='true' [filterSettings]='filterOption'>
    <e-columns>
        <e-column field='TaskID' headerText='Task ID' width='70'
textAlign='Right' [filter]='filterParams'></e-column>
        <e-column field='TaskName' headerText='Task Name' width='100'
></e-column>
        <e-column field='StartDate' headerText='Start Date'
textAlign='Right' [format]='formatOptions' width='100' ></e-column>
        <e-column field='EndDate' headerText='End Date'
textAlign='Right' [format]='formatOptions' width='100'></e-column>
        <e-column field='Duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
        <e-column field='Priority' headerText='Priority' width='90'></e-
column>
    </e-columns>
    </ejs-treegrid>`,
  })
export class AppComponent implements OnInit {
  public data: Object[] = [];
  public formatOptions?: Object;
  public filterParams?: IFilter;
  public filterOption?: FilterSettingsModel;
  ngOnInit(): void {
    this.data = projectData;
    this.formatOptions = { format: 'y/M/d', type: 'date' };
    this.filterOption = { type: 'Menu' };
    this.filterParams = { params: { showSpinButton: false } };
  }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Select treegrid rows based on certain condition in Angular Treegrid component

You can select the specific row in the Tree Grid based on a certain condition by using the [selectRows](#) method in the [dataBound](#) event of Tree Grid.

In the below demo, we have selected the Tree Grid rows only when *Duration* column value greater than 4.

APP.COMPONENT.TS

```

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

```

```

import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent, SelectionSettingsModel } from '@syncfusion/ej2-
angular-treegrid';
import { RowDataBoundEventArgs } from '@syncfusion/ej2-angular-grids';
import { getValue } from '@syncfusion/ej2-base';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treegrid #treegridObj [dataSource]='data'
idMapping='TaskID' parentIdMapping='parentID'
[treeColumnIndex]='1' [height]='269' allowPaging='true'
(rowDataBound)='rowDataBound($event)'
(dataBound)='dataBound($event)' [selectionSettings]='selectionOptions'>
    <e-columns>
      <e-column field='TaskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
      <e-column field='TaskName' headerText='Task Name' width='100'
></e-column>
      <e-column field='StartDate' headerText='Start Date'
textAlign='Right' [format]='formatOptions' editType='datepickeredit'
width='100' ></e-column>
      <e-column field='EndDate' headerText='End Date'
textAlign='Right' [format]='formatOptions' editType='datepickeredit'
width='100'></e-column>
      <e-column field='Duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
      <e-column field='Priority' headerText='Priority' width='90'></e-
column>
    </e-columns>
  </ejs-treegrid>`,
})
export class AppComponent implements OnInit {
  public data: Object[] = [];
  public formatOptions?: Object;
  public selectionOptions?: SelectionSettingsModel;
  public selIndex: number[] = [];
  @ViewChild('treegridObj')
  public treegridObj?: TreeGridComponent;
  ngOnInit(): void {
    this.data = projectData;
    this.formatOptions = { format: 'y/M/d', type: 'date' };
    this.selectionOptions = { type: 'Multiple' };
  }
}

```

```

    }
    rowDataBound(args: RowDataBoundEventArgs) {
    if (getValue('Duration', args.data as object) > 4) {
        this.selIndex.push(parseInt((args.row as HTMLTableRowElement)
            .getAttribute('aria-rowindex') as string, 0));
    }
    }
    dataBound(args: any) {
        if (this.treegridObj && this.selIndex.length) {
            this.treegridObj.selectRows(this.selIndex);
            this.selIndex = [];
        }
    }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Get row cell index in Angular Treegrid component

You can get the specific row and cell index of the Tree Grid by using [rowSelected](#) event of the treegrid. Here, we can get the row and cell index by using *aria-rowindex* (get row Index from *tr* element) and *aria-colindex* (column index from *td* element) attribute.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { projectData } from './datasource';
import { TreeGridComponent } from '@syncfusion/ej2-angular-treegrid';
import { RowSelectEventArgs } from '@syncfusion/ej2-angular-grids';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',

```

```

    template: `<ejs-treegrid #treegridObj [dataSource]='data'
idMapping='TaskID' parentIdMapping='parentID' [treeColumnIndex]='1'
(rowSelected)='rowSelected($event)' [height]='267'>
    <e-columns>
        <e-column field='TaskID' headerText='Task ID' width='70'
textAlign='Right'></e-column>
        <e-column field='TaskName' headerText='Task Name' width='150'
></e-column>
        <e-column field='StartDate' headerText='Start Date'
textAlign='Right' [format]='formatOptions' width='90'></e-column>
        <e-column field='EndDate' headerText='Start Date'
textAlign='Right' [format]='formatOptions' width='90'></e-column>
        <e-column field='Duration' headerText='Duration' width='80'
textAlign='Right'></e-column>
        <e-column field='Progress' headerText='Progress' width='80'
textAlign='Right'></e-column>
        <e-column field='Priority' headerText='Priority' width='90'></e-
column>
    </e-columns>
</ejs-treegrid>`,
})
export class AppComponent implements OnInit {
    public data: Object[] = [];
    public formatOptions?: Object;
    ngOnInit(): void {
        this.data = projectData;
        this.formatOptions = { format: 'y/M/d', type: 'date' };
    }
    rowSelected(args: RowSelectEventArgs) {
        alert("row index: " + " " + (args.row as
HTMLTableRowElement).getAttribute('aria-rowindex'));
        alert("column index: " + " " + ((args.target as
Element).closest('td') as HTMLTableCellElement).getAttribute('aria-
colindex'));
    }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Display foreign key values in treegrid in Angular Treegrid component

Since Tree Grid Databinding concept is of hierarchy relationship, we do not provide in-built support for foreignKey datasource.

To display the foreignKey value at initial rendering, we can use the [queryCellInfo](#) event of the Tree Grid and also by using the [editType](#) and [columns.edit](#) properties of Tree Grid Column, we can render Dropdownlist with external or foreign dataSource.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { foreignKeyData, dropData } from './datasource';
import { TreeGridComponent, EditService, ToolbarService, ITreeData,
EditSettingsModel, ToolbarItems } from '@syncfusion/ej2-angular-treegrid';
import { QueryCellInfoEventArgs, Column, IEditCell } from '@syncfusion/ej2-
angular-grids';
import { DataManager, Query } from '@syncfusion/ej2-data';
@Component({
  imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule
  ],
  providers: [PageService,
    SortService,
    FilterService],
  standalone: true,
  selector: 'app-container',
  providers: [ToolbarService, EditService],
  template: `<ejs-treegrid #treegridObj [dataSource]='data'
childMapping='Children' [treeColumnIndex]='1' [editSettings]='editSettings'
[toolbar]='toolbar' (queryCellInfo)='queryCellInfo($event)' [height]='273' >
    <e-columns>
      <e-column field='EmpID' headerText='EmpID' width='70'
textAlign='Right' isPrimaryKey=true ></e-column>
      <e-column field='Name' headerText='Employee Name' width='70'
></e-column>
      <e-column field='Contact' headerText='Contact' width='90'
textAlign='Right'></e-column>
      <e-column field='DOB' headerText='DOB' textAlign='Right'
[format]='formatOptions' editType='datepickeredit' width='70'></e-column>
      <e-column field='EmployeeID' headerText='Employee ID' editType='
dropdownedit' [edit]='employeeParams' width='70'></e-column>
      <e-column field='Country' headerText='Country' width='90'
textAlign='Right'></e-column>
    </e-columns>
  </ejs-treegrid>`,
})
export class AppComponent implements OnInit {
  public data: Object[] = [];
  public formatOptions?: Object;
  public editSettings?: EditSettingsModel;
  public toolbar?: ToolbarItems[];
  public employeeParams?: IEditCell;
  ngOnInit(): void {
    this.data = foreignKeyData;
    this.formatOptions = { format: 'y/M/d', type: 'date' };
  }
}

```

```

        this.editSettings = { allowEditing: true, allowAdding: true,
allowDeleting: true, mode: 'Cell' };
        this.toolbar = ['Add', 'Edit', 'Delete', 'Update', 'Cancel'];
        // Bind ForeignKey DataSource for dropdown using editParams
        this.employeeParams = {
            params: {
                dataSource: new DataManager(dropData),
                fields: { text: "EmployeeName", value: "EmployeeID" },
                query: new Query()
            }
        };
    }
    queryCellInfo(args: QueryCellInfoEventArgs | any) {
        if ((args.column as Column).field === "EmployeeID") {
            for (var i = 0; i < dropData.length; i++) {
                let data: Object[] = args.data as Object[];
                if (data[(args.column).field] === (dropData as
any)[i]["EmployeeID"]) {
                    (args.cell as HTMLElement).innerText = (dropData as
any)[i]["EmployeeName"]; // assign the foreignkey field value to the
innertext
                }
            }
        }
    }
}

```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

Row cell customization in Angular Treegrid component

In Tree Grid we can customize the row and cell using [queryCellInfo](#) and [rowDataBound](#) events of Tree Grid.

In the below demo, we customize and show the command buttons only for the parent rows using [queryCellInfo](#) and [rowDataBound](#) events of Tree Grid.

APP.COMPONENT.TS

```

import { NgModule, ViewChild } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeGridModule } from '@syncfusion/ej2-angular-treegrid'
import { PageService, SortService, FilterService } from '@syncfusion/ej2-
angular-treegrid'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild } from '@angular/core';
import { sampleData } from './datasource';

```



```

import { TreeGridComponent, CommandColumnService, ITreeData } from
'@syncfusion/ej2-angular-treegrid';
import { QueryCellInfoEventArgs, RowDataBoundEventArgs } from
'@syncfusion/ej2-angular-grids';
@Component({
imports: [

    TreeGridModule,
    ButtonModule,
    DropDownListAllModule

],
providers: [PageService,
            SortService,
            FilterService],
standalone: true,
selector: 'app-container',
providers: [ CommandColumnService ],
template: `<ejs-treegrid #treegridObj [dataSource]='data'
childMapping='subtasks'
[treeColumnIndex]='1' (rowDataBound)='rowDataBound($event)'
(queryCellInfo)='queryCellInfo($event)' [height]='280' >
    <e-columns>
        <e-column field='taskID' headerText='Task ID' width='80'
textAlign='Right'></e-column>
        <e-column field='taskName' headerText='Task Name' width='200'
></e-column>
        <e-column field='startDate' headerText='Start Date'
textAlign='Right' [format]='formatOptions' width='140'></e-column>
        <e-column field='endDate' headerText='Start Date'
textAlign='Right' [format]='formatOptions' width='140'></e-column>
        <e-column field='duration' headerText='Duration' width='90'
textAlign='Right'></e-column>
        <e-column field='progress' headerText='Progress' width='90'
textAlign='Right'></e-column>
        <e-column headerText='Custom Button' [commands]='commands'
width='160' textAlign='Right'></e-column>
    </e-columns>
</ejs-treegrid>`,
})
export class AppComponent implements OnInit {
    public data: Object[] = [];
    public formatOptions?: Object;
    public commands?: Object[];
    ngOnInit(): void {
        this.data = sampleData;
        this.formatOptions = { format: 'y/M/d', type: 'date' };
        this.commands = [{ buttonOption: { content: 'Details', cssClass: 'e-
flat', click: onclick } }];
    }
    queryCellInfo(args: QueryCellInfoEventArgs) {
        if (!(args.data as ITreeData).hasChildRecords){
            if ((args.cell as HTMLElement).classList.contains("e-
unboundcell")) {
                ((args.cell as HTMLElement).querySelector('.e-
unboundcelldiv') as HTMLElement).style.display = "none";
            }
        }
    }
}

```

```
}  
rowDataBound(args: RowDataBoundEventArgs) {  
    if (!args.data as ITreeData).hasChildRecords) {  
        (args.row as HTMLElement).style.backgroundColor = 'green';  
    }  
}  
}
```

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 Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tree Grid example](#) to know how to present and manipulate data.

TreeMap

Getting started with Angular Treemap component

This section explains you the steps required to create a TreeMap control and demonstrate the basic usage of the TreeMap 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
```

```
,
```

Adding Syncfusion TreeMap package

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) registry.

To install treemap component, use the following command.

```
`bash
```

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

```
,
```

The `--save` will instruct NPM to include the treemap package inside of the `dependencies` section of the `package.json`.

Registering TreeMap Module

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

```
`typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// import the TreeMapModule for the TreeMap component
import { TreeMapModule } from '@syncfusion/ej2-angular-treemap';
import { AppComponent } from './app.component';
@NgModule({
//declaration of chart module into NgModule
imports: [ BrowserModule, TreeMapModule ],
declarations: [ AppComponent ],
bootstrap: [ AppComponent ]
})
export class AppModule { }
```

- Modify the template in `app.component.ts` file to render the `ej2-angular-treemap` component [`src/app/app.component.ts`].

```
`javascript
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
selector: 'app-container',
// specifies the template string for the treemap component
template: <ejs-treemap id='treemap-container'></ejs-treemap>,
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 treemap.

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

@Component({
  selector: 'app-container',
  // specifies the template string for the treemap component
  template: <ejs-treemap id="treemap-container"></ejs-treemap>
})
export class AppComponent {
}
```

Since we did not specify [dataSource](#) for the TreeMap, no items will be rendered and only an empty SVG element will be appended to the treemap container.

Render TreeMap

This section explains how to render a TreeMap with a data source. In this example, we are going to modify the above basic treemap to visualize international airport count in South America.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;
height='350px' [dataSource]='data' weightValuePath='Count'
[leafItemSettings]='leafItemSettings'>`
})
export class AppComponent {
}
```

```

    </ejs-treemap>`
  })
  export class AppComponent {
    public data: object[] = [
      { Title: 'State wise International Airport count in South America',
        State: 'Brazil', Count: 25 },
      { Title: 'State wise International Airport count in South America',
        State: 'Colombia', Count: 12 },
      { Title: 'State wise International Airport count in South America',
        State: 'Argentina', Count: 9 },
      { Title: 'State wise International Airport count in South America',
        State: 'Ecuador', Count: 7 },
      { Title: 'State wise International Airport count in South America',
        State: 'Chile', Count: 6 },
      { Title: 'State wise International Airport count in South America',
        State: 'Peru', Count: 3 },
      { Title: 'State wise International Airport count in South America',
        State: 'Venezuela', Count: 3 },
      { Title: 'State wise International Airport count in South America',
        State: 'Bolivia', Count: 2 },
      { Title: 'State wise International Airport count in South America',
        State: 'Paraguay', Count: 2 },
      { Title: 'State wise International Airport count in South America',
        State: 'Uruguay', Count: 2 },
      { Title: 'State wise International Airport count in South America',
        State: 'Falkland Islands', Count: 1 },
      { Title: 'State wise International Airport count in South America',
        State: 'French Guiana', Count: 1 },
      { Title: 'State wise International Airport count in South America',
        State: 'Guyana', Count: 1 },
      { Title: 'State wise International Airport count in South America',
        State: 'Suriname', Count: 1 },
    ];
    public leafItemSettings: object = {
      labelPath: 'State'
    };
  }
}

```

MAIN.TS

```

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

```

Here, TreeMap is displayed using a data source, and the [weightValuePath](#) property is set to the data source's **Count** field as the value. The leaf level items of TreeMap can be customized using [leafItemSettings](#). `leafItemSettings` allows you to change properties such as [fill](#), [border](#) and [labelPosition](#).

Module Injection

The TreeMap component is divided into individual feature-based modules. To use a specific feature, you must inject its **Service** module into the `AppModule`. The modules available in TreeMap, as well as their descriptions, are listed below.

- TreeMapHighlightService - Inject this provider to use highlight feature.
- TreeMapSelectionService - Inject this provider to use selection feature.
- TreeMapLegendService - Inject this provider to use legend feature.
- TreeMapTooltipService - Inject this provider to use tooltip feature.

Apply Color Mapping

The color mapping feature supports customization of item colors based on the underlying value of item received from bounded datasource. Specify the field name from which the values have to be compared for the item in [equalColorValuePath](#) or [rangeColorValuePath](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
height='350px' [dataSource]='data' [legendSettings]='legendSettings'
equalColorValuePath='Count' weightValuePath='Count'
[leafItemSettings]='leafItemSettings'>
  </ejs-treemap>`
})
export class AppComponent {
  public legendSettings: object = {
    visible: true,
  };
  public data: object[] = [
    { Title: 'State wise International Airport count in South America',
State: 'Brazil', Count: 25 },
    { Title: 'State wise International Airport count in South America',
State: 'Colombia', Count: 12 },
    { Title: 'State wise International Airport count in South America',
State: 'Argentina', Count: 9 },
    { Title: 'State wise International Airport count in South America',
State: 'Ecuador', Count: 7 },
    { Title: 'State wise International Airport count in South America',
State: 'Chile', Count: 6 },
    { Title: 'State wise International Airport count in South America',
State: 'Peru', Count: 3 },
    { Title: 'State wise International Airport count in South America',
State: 'Venezuela', Count: 3 },
    { Title: 'State wise International Airport count in South America',
State: 'Bolivia', Count: 2 },
    { Title: 'State wise International Airport count in South America',
State: 'Paraguay', Count: 2 },
    { Title: 'State wise International Airport count in South America',
State: 'Uruguay', Count: 2 },
```

```

    { Title: 'State wise International Airport count in South America',
      State: 'Falkland Islands', Count: 1 },
    { Title: 'State wise International Airport count in South America',
      State: 'French Guiana', Count: 1 },
    { Title: 'State wise International Airport count in South America',
      State: 'Guyana', Count: 1 },
    { Title: 'State wise International Airport count in South America',
      State: 'Suriname', Count: 1 },
  ];
  public leafItemSettings: object = {
    labelPath: 'State',
    colorMapping: [
      {
        value: 25,
        color: '#634D6F'
      },
      {
        value: 12,
        color: '#B34D6D'
      },
      {
        value: 9,
        color: '#557C5C'
      },
      {
        value: 7,
        color: '#44537F'
      },
      {
        value: 6,
        color: '#637392'
      },
      {
        value: 3,
        color: '#7C754D'
      },
      {
        value: 2,
        color: '#2E7A64'
      },
      {
        value: 1,
        color: '#95659A'
      }
    ]
  };
}

```

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 show legend for the TreeMap by setting true to the [visible](#) property in [legendSettings](#) object and by injecting the [TreeMapLegendService](#) module in the AppModule.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapAllModule } from
 '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[legendSettings]='legendSettings' [dataSource]='data'
equalColorValuePath='Count' weightValuePath='Count'
[leafItemSettings]='leafItemSettings'>
    </ejs-treemap>`
})
export class AppComponent {
  public legendSettings: object = {
    visible: true,
    position: 'Top',
    shape: 'Rectangle'
  };
  public data: object[] = [
    { Title: 'State wise International Airport count in South America',
State: 'Brazil', Count: 25 },
    { Title: 'State wise International Airport count in South America',
State: 'Colombia', Count: 12 },
    { Title: 'State wise International Airport count in South America',
State: 'Argentina', Count: 9 },
    { Title: 'State wise International Airport count in South America',
State: 'Ecuador', Count: 7 },
    { Title: 'State wise International Airport count in South America',
State: 'Chile', Count: 6 },
    { Title: 'State wise International Airport count in South America',
State: 'Peru', Count: 3 },
    { Title: 'State wise International Airport count in South America',
State: 'Venezuela', Count: 3 },
    { Title: 'State wise International Airport count in South America',
State: 'Bolivia', Count: 2 },
    { Title: 'State wise International Airport count in South America',
State: 'Paraguay', Count: 2 },
    { Title: 'State wise International Airport count in South America',
State: 'Uruguay', Count: 2 },
    { Title: 'State wise International Airport count in South America',
State: 'Falkland Islands', Count: 1 },
    { Title: 'State wise International Airport count in South America',
State: 'French Guiana', Count: 1 },
    { Title: 'State wise International Airport count in South America',
State: 'Guyana', Count: 1 },
```



```

    { Title: 'State wise International Airport count in South America',
      State: 'Suriname', Count: 1 },
    ];
    public leafItemSettings: object = {
      labelPath: 'State',
      colorMapping: [
        {
          value: 25,
          color: '#634D6F'
        },
        {
          value: 12,
          color: '#B34D6D'
        },
        {
          value: 9,
          color: '#557C5C'
        },
        {
          value: 7,
          color: '#44537F'
        },
        {
          value: 6,
          color: '#637392'
        },
        {
          value: 3,
          color: '#7C754D'
        },
        {
          value: 2,
          color: '#2E7A64'
        },
        {
          value: 1,
          color: '#95659A'
        }
      ]
    };
  }
}

```

MAIN.TS

```

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

```

Add Labels

Labels can be added to show additional information of the items in TreeMap. By default, visibility of the label is true. This can be customized using [showLabels](#) property in [leafItemSettings](#).

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapAllModule } from
 '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[legendSettings]='legendSettings' [dataSource]='data'
equalColorValuePath='Count' weightValuePath='Count'
[leafItemSettings]='leafItemSettings'>
    </ejs-treemap>`
})
export class AppComponent {
  public legendSettings: object = {
    visible: true,
    position: 'Top',
    shape: 'Rectangle'
  };
  public data: object[] = [
    { Title: 'State wise International Airport count in South America',
State: 'Brazil', Count: 25 },
    { Title: 'State wise International Airport count in South America',
State: 'Colombia', Count: 12 },
    { Title: 'State wise International Airport count in South America',
State: 'Argentina', Count: 9 },
    { Title: 'State wise International Airport count in South America',
State: 'Ecuador', Count: 7 },
    { Title: 'State wise International Airport count in South America',
State: 'Chile', Count: 6 },
    { Title: 'State wise International Airport count in South America',
State: 'Peru', Count: 3 },
    { Title: 'State wise International Airport count in South America',
State: 'Venezuela', Count: 3 },
    { Title: 'State wise International Airport count in South America',
State: 'Bolivia', Count: 2 },
    { Title: 'State wise International Airport count in South America',
State: 'Paraguay', Count: 2 },
    { Title: 'State wise International Airport count in South America',
State: 'Uruguay', Count: 2 },
    { Title: 'State wise International Airport count in South America',
State: 'Falkland Islands', Count: 1 },
    { Title: 'State wise International Airport count in South America',
State: 'French Guiana', Count: 1 },
    { Title: 'State wise International Airport count in South America',
State: 'Guyana', Count: 1 },
    { Title: 'State wise International Airport count in South America',
State: 'Suriname', Count: 1 },
  ];
  public leafItemSettings: object = {
    showLabels: true,
    labelPath: 'State',

```

```
    labelPosition: 'Center',
    labelStyle: {
      color: 'white'
    },
    colorMapping: [
      {
        value: 25,
        color: '#634D6F'
      },
      {
        value: 12,
        color: '#B34D6D'
      },
      {
        value: 9,
        color: '#557C5C'
      },
      {
        value: 7,
        color: '#44537F'
      },
      {
        value: 6,
        color: '#637392'
      },
      {
        value: 3,
        color: '#7C754D'
      },
      {
        value: 2,
        color: '#2E7A64'
      },
      {
        value: 1,
        color: '#95659A'
      }
    ]
  };
}
```

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 labels cannot display information by using due to space constraints. Tooltip can be enabled by setting the [visible](#) property as true in [tooltipSettings](#) object and by injecting `TreeMapTooltipService` module in the AppModule.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[tooltipSettings]='tooltipSettings' [legendSettings]='legendSettings'
[dataSource]='data' equalColorValuePath='Count' weightValuePath='Count'
[leafItemSettings]='leafItemSettings'>
    </ejs-treemap>`
})
export class AppComponent {
  public legendSettings: object = {
    visible: true,
    position: 'Top',
    shape: 'Rectangle'
  };
  public tooltipSettings: object = {
    visible: true,
  };
  public data: object[] = [
    { Title: 'State wise International Airport count in South America',
State: 'Brazil', Count: 25 },
    { Title: 'State wise International Airport count in South America',
State: 'Colombia', Count: 12 },
    { Title: 'State wise International Airport count in South America',
State: 'Argentina', Count: 9 },
    { Title: 'State wise International Airport count in South America',
State: 'Ecuador', Count: 7 },
    { Title: 'State wise International Airport count in South America',
State: 'Chile', Count: 6 },
    { Title: 'State wise International Airport count in South America',
State: 'Peru', Count: 3 },
    { Title: 'State wise International Airport count in South America',
State: 'Venezuela', Count: 3 },
    { Title: 'State wise International Airport count in South America',
State: 'Bolivia', Count: 2 },
    { Title: 'State wise International Airport count in South America',
State: 'Paraguay', Count: 2 },
    { Title: 'State wise International Airport count in South America',
State: 'Uruguay', Count: 2 },
    { Title: 'State wise International Airport count in South America',
State: 'Falkland Islands', Count: 1 },
    { Title: 'State wise International Airport count in South America',
State: 'French Guiana', Count: 1 },
    { Title: 'State wise International Airport count in South America',
State: 'Guyana', Count: 1 },
    { Title: 'State wise International Airport count in South America',
State: 'Suriname', Count: 1 },
  ];
};

```

```
public leafItemSettings: object = {
  showLabels: true,
  labelPath: 'State',
  labelPosition: 'Center',
  labelStyle: {
    color: 'white'
  },
  colorMapping: [
    {
      value: 25,
      color: '#634D6F'
    },
    {
      value: 12,
      color: '#B34D6D'
    },
    {
      value: 9,
      color: '#557C5C'
    },
    {
      value: 7,
      color: '#44537F'
    },
    {
      value: 6,
      color: '#637392'
    },
    {
      value: 3,
      color: '#7C754D'
    },
    {
      value: 2,
      color: '#2E7A64'
    },
    {
      value: 1,
      color: '#95659A'
    }
  ]
};
```

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 Treemap component

The TreeMap control supports data binding using the dataSource property.

Populate data

The [dataSource](#) property accepts collection values as input. For example, a list of objects can be provided as input. Data can be given as either flat or hierarchical collection to the [dataSource](#) property.

<!-- markdownlint-disable MD036 -->

Flat collection

The following code shows, how to bind a flat collection as data source to the TreeMap control.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
height='350px' [dataSource]='data' weightValuePath='GDP'
[leafItemSettings]='leafItemSettings'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},
    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
  ];
  public leafItemSettings: object = {
    labelPath: 'State'
  };
}
```

MAIN.TS

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

Layout in Angular Treemap component

Determine the visual representation of nodes belonging to all the TreeMap levels using the [layoutType](#) property.

Types of layout

The available layout types are,

- Squarified
- SliceAndDiceVertical
- SliceAndDiceHorizontal
- SliceAndDiceAuto

Squarified

The **Squarified** layout displays the nested rectangles based on aspect ratio in the TreeMap. The rectangles will be split based on the height and width of the parent. The default rendering type of layout is **Squarified**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='GDP'
>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},
    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
  ];
}
```

MAIN.TS

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

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

SliceAndDiceVertical

The **SliceAndDiceVertical** layout creates rectangles with high aspect ratio and displays items in a vertically sorted order.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='GDP' layoutType= 'SliceAndDiceVertical'
>
    </ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},
    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
  ];
}
```

MAIN.TS

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

SliceAndDiceHorizontal

The **SliceAndDiceHorizontal** layout creates rectangles with high aspect ratio and displays items in a horizontally sorted order.

APP.COMPONENT.TS


```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='GDP' layoutType=
'SliceAndDiceHorizontal'
>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},
    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
  ];
}
```

MAIN.TS

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

SliceAndDiceAuto

The **SliceAndDiceAuto** layout creates rectangles with high aspect ratio and display items sorted both horizontally and vertically.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
```

```

providers: [TreeMapLegendService, TreeMapTooltipService],
standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='GDP' layoutType= 'SliceAndDiceAuto'
>
  </ejs-treemap>`
}))
export class AppComponent {
  public data: object[] = [
    {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},
    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
  ];
}

```

MAIN.TS

```

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

```

Leaf item in Angular Treemap component

A leaf item defines a visualized data element and does not contain child nodes but contains parent node if the levels are specified in the TreeMap.

Leaf label

Label is represented by item name or value. Label will be appeared by specifying the [labelPath](#) property and customize the label style using the [labelStyle](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='GDP'
[leafItemSettings]='leafItemSettings' >

```

```

    </ejs-treemap>`
  })
  export class AppComponent {
    public data: object[] = [
      {State:"United States", GDP:17946, percentage:11.08, Rank:1},
      {State:"China", GDP:10866, percentage: 28.42, Rank:2},
      {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
      {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
      {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
      {State:"France", GDP:2421, percentage:-9.69, Rank:6},
      {State:"India", GDP:2073, percentage:13.65, Rank:7},
      {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
      {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
      {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
    ];
    public leafItemSettings: object= {
      labelPath: 'State',
      labelStyle: {
        color: '#000000'
      },
      border: {
        color: '#000000',
        width: 0.5
      }
    }
  }
}

```

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 -->

Label position and format

Positioning the leaf item label using the [labelPosition](#) property and the text format can be customized by specifying data source properties name in the [labelFormat](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='GDP'

```

```

    [leafItemSettings]='leafItemSettings' >
  </ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},
    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
  ];
  public leafItemSettings: object= {
    labelPath: 'State',
    labelPosition: 'TopCenter',
    labelFormat: '$ ${State}<br>$$ {GDP} Trillion<br>($ {percentage} %)',
  };
}

```

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 -->

Label template and position

Specifies the template of leaf item label and position of the template to be customized using [labelTemplate](#) and [templatePosition](#) properties.

`typescript

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

```
@Component({
```

```
  selector: 'app-container',
```

```
  template: `<ejs-treemap id='container' style='display: block;' [dataSource]='data'
  weightValuePath='Gold'`
```

```
  [leafItemSettings]='leafItemSettings' >
```

```
</ejs-treemap>`
```

```
})
```

```
export class AppComponent {
```

```
  public data: object[] = [
```

```
    { Sport: "Swimming", Gold: 16, GameImage: 'Swimming.svg', ItemHeight: "180px", ItemWidth: '180px' },
```

```
{ Sport: "Athletics", Gold: 13, GamelImage: 'Athletics.svg', ItemHeight: "70px", ItemWidth: '70px' },
{ Sport: "Gymnastics", Gold: 4, GamelImage: 'Gymnastics.svg', ItemHeight: "80px", ItemWidth: '80px' },
{ Sport: "Cycling", Gold: 2, GamelImage: 'Cycling.svg', ItemHeight: "50px", ItemWidth: '50px' },
{ Sport: "Wrestling", Gold: 2, GamelImage: 'Wrestling.svg', ItemHeight: "60px", ItemWidth: '50px' },
{ Sport: "Basketball", Gold: 2, GamelImage: 'Basketball.svg', ItemHeight: "50px", ItemWidth: '50px' },
{ Sport: "Boxing", Gold: 1, GamelImage: 'Boxing.svg', ItemHeight: "40px", ItemWidth: '30px' },
{ Sport: "Tennis", Gold: 1, GamelImage: 'Tennis.svg', ItemHeight: "40px", ItemWidth: '40px' },
{ Sport: "Judo", Gold: 1, GamelImage: 'Judo.svg', ItemHeight: "40px", ItemWidth: '40px' },
{ Sport: "Rowing", Gold: 1, GamelImage: 'Rowing.svg', ItemHeight: "40px", ItemWidth: '40px' },
{ Sport: "Shooting", Gold: 1, GamelImage: 'Shooting.svg', ItemHeight: "40px", ItemWidth: '40px' },
{ Sport: "Triathlon", Gold: 1, GamelImage: 'Triathlon.svg', ItemHeight: "40px", ItemWidth: '40px' },
{ Sport: "Water polo", Gold: 1, GamelImage: 'Water polo.svg', ItemHeight: "40px", ItemWidth: '40px' }
];

public leafItemSettings: object = {
  labelPath: 'Sport',
  fill: '#993399',
  templatePosition: 'Center',
  labelTemplate: '<div style="pointer-events: none;"></img></div>'
}
}
```

<!-- markdownlint-disable MD036 -->

Item gap

The [gap](#) property is used to separate an item from another item. Each item rectangle is split into equal space with specified gap.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
```

```

    template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='GDP'
    [leafItemSettings]='leafItemSettings' >
    </ejs-treemap>`
  })
  export class AppComponent {
    public data: object[] = [
      {State:"United States", GDP:17946, percentage:11.08, Rank:1},
      {State:"China", GDP:10866, percentage: 28.42, Rank:2},
      {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
      {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
      {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
      {State:"France", GDP:2421, percentage:-9.69, Rank:6},
      {State:"India", GDP:2073, percentage:13.65, Rank:7},
      {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
      {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
      {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
    ];
    public leafItemSettings: object= {
      labelPath: 'State',
      gap:20
    };
  }

```

MAIN.TS

```

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

```

Levels in Angular Treemap component

TreeMap supports **n** number of levels and each level is separated by using the [groupPath](#) property.

<!-- markdownlint-disable MD036 -->

Group path

The [groupPath](#) property is used to separate each level of the TreeMap by specifying the property from the data source.

In the following example, three levels are added and each level is configured using the [groupPath](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,

```

```

        selector: 'app-container',
        template: `<ejs-treemap id='container' style='display: block;'
height='350px' [dataSource]='data' weightValuePath='EmployeesCount'
[palette]='palette'
[levels]='levels'>
        </ejs-treemap>`
    })
    export class AppComponent {
        public data: object[] = [
            { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 20 },
            { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 30 },
            { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
EmployeesCount: 40 },
            { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
EmployeesCount: 80 },
            { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 100 },
            { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
            { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
EmployeesCount: 40 },
            { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 50 },
            { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 60 },
            { Category: 'Employees', Country: 'Germany', JobDescription: 'Marketing',
EmployeesCount: 70 },
            { Category: 'Employees', Country: 'Germany', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 80 },
            { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
            { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
EmployeesCount: 20 },
            { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 30 },
            { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
EmployeesCount: 50 },
            { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
EmployeesCount: 60 },
            { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 70 },
            { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
EmployeesCount: 100 }
        ];
        public levels: object[] = [
            { groupPath: 'Country', border: { color: 'black', width: 0.5 } },
            { groupPath: 'JobDescription', border: { color: 'black', width: 0.5 } },
        ],
        { groupPath: 'JobGroup', border: { color: 'black', width: 0.5 } },
    ]
        public palette: object = ["#f44336", "#29b6f6", "#ab47bc", "#ffc107",
"#5c6bc0", "#009688"];
    }

```

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 -->

Group gap

The [groupGap](#) property is used to separate an item from each group or another item to differentiate the levels mentioned in the TreeMap.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
height='350px' [dataSource]='data' weightValuePath='EmployeesCount'
[palette]='palette'
[levels]='levels'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 20 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
EmployeesCount: 80 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 100 },
    { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 50 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 60 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Marketing',
EmployeesCount: 70 },
```



```

    { Category: 'Employees', Country: 'Germany', JobDescription: 'Technical',
      JobGroup: 'Testers', EmployeesCount: 80 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
      'Management', EmployeesCount: 10 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
      EmployeesCount: 20 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
      JobGroup: 'Testers', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
      EmployeesCount: 50 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
      EmployeesCount: 60 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
      JobGroup: 'Testers', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
      EmployeesCount: 100 }
  ];
  public levels: object[] = [
    { groupPath: 'Country', groupGap: 10, border: { color: 'black', width:
0.5 } },
    { groupPath: 'JobDescription', groupGap: 10, border: { color: 'black',
width: 0.5 } },
    { groupPath: 'JobGroup', groupGap: 10, border: { color: 'black',
width: 0.5 } },
  ]
  public palette: object = ["#f44336", "#29b6f6", "#ab47bc", "#ffc107",
"#5c6bc0", "#009688"];
}

```

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 -->

Header format and Alignment

Customize header using the [headerFormat](#) property in which fields are mapping from the dataSource and align header using the [headerAlignment](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',

```

```

    template: `<ejs-treemap id='container' style='display: block;'
height='350px' [dataSource]='data' weightValuePath='EmployeesCount'
[palette]='palette'
    [levels]='levels'>
    </ejs-treemap>`
  })
  export class AppComponent {
    public data: object[] = [
      { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 20 },
      { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 30 },
      { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
EmployeesCount: 40 },
      { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
EmployeesCount: 80 },
      { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 100 },
      { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
      { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
EmployeesCount: 40 },
      { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 50 },
      { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 60 },
      { Category: 'Employees', Country: 'Germany', JobDescription: 'Marketing',
EmployeesCount: 70 },
      { Category: 'Employees', Country: 'Germany', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 80 },
      { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
      { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
EmployeesCount: 20 },
      { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 30 },
      { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
EmployeesCount: 50 },
      { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
EmployeesCount: 60 },
      { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 70 },
      { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
EmployeesCount: 100 }
    ];
    public levels: object[] = [
      { groupPath: 'Country', headerFormat: '${Country}-${EmployeesCount}',
headerAlignment: 'Center', border: { color: 'black', width: 0.5 } },
      { groupPath: 'JobDescription', headerFormat: '${JobDescription}-
${EmployeesCount}', headerAlignment: 'Far', border: { color: 'black', width:
0.5 } },
      { groupPath: 'JobGroup', headerAlignment: 'Near',
headerFormat: '${JobGroup}-${EmployeesCount}', border: { color:
'black', width: 0.5 } },
    ]
    public palette: object = ["#f44336", "#29b6f6", "#ab47bc", "#ffc107",
"#5c6bc0", "#009688"];
  }

```

```
}

```

MAIN.TS

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

Header height and style

Customize the font color, family, weight, opacity and size using the [headerStyle](#). Based on the font settings, the header height is given using the [headerHeight](#) property in [levels](#).

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
height='350px' [dataSource]='data' weightValuePath='EmployeesCount'
[palette]='palette'
[levels]='levels'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 20 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
EmployeesCount: 80 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 100 },
    { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 50 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 60 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Marketing',
EmployeesCount: 70 },
```

```

    { Category: 'Employees', Country: 'Germany', JobDescription: 'Technical',
      JobGroup: 'Testers', EmployeesCount: 80 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
      'Management', EmployeesCount: 10 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
      EmployeesCount: 20 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
      JobGroup: 'Testers', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
      EmployeesCount: 50 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
      EmployeesCount: 60 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
      JobGroup: 'Testers', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
      EmployeesCount: 100 }
  ];
  public levels: object[] = [
    { groupPath: 'Country', headerHeight: 35, headerStyle: { size: '15px' },
      border: { color: 'black', width: 0.5 } },
    { groupPath: 'JobDescription', headerHeight: 45, headerStyle: {
      size: '15px' }, border: { color: 'black', width: 0.5 } },
    { groupPath: 'JobGroup', headerHeight: 40, headerStyle: { size: '15px' },
      border: { color: 'black', width: 0.5 } },
  ]
  public palette: object = ["#f44336", "#29b6f6", "#ab47bc", "#ffc107",
    "#5c6bc0", "#009688"];
}

```

MAIN.TS

```

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

```

Header template and position

The TreeMap header supports to customize header of each item using the [headerTemplate](#) property. It uses Essential JS2 Template engine to render the elements. You can position the template using the [templatePosition](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
  TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',

```

```

    template: `<ejs-treemap id='container' style='display: block;'
height='350px' [dataSource]='data' weightValuePath='EmployeesCount'
[palette]='palette'
    [levels]='levels'>
    </ejs-treemap>`
  })
  export class AppComponent {
    public data: object[] = [
      { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 20 },
      { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 30 },
      { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
EmployeesCount: 40 },
      { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
EmployeesCount: 80 },
      { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 100 },
      { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
      { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
EmployeesCount: 40 },
      { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 50 },
      { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 60 },
      { Category: 'Employees', Country: 'Germany', JobDescription: 'Marketing',
EmployeesCount: 70 },
      { Category: 'Employees', Country: 'Germany', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 80 },
      { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
      { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
EmployeesCount: 20 },
      { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 30 },
      { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
EmployeesCount: 50 },
      { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
EmployeesCount: 60 },
      { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 70 },
      { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
EmployeesCount: 100 }
    ];
    public levels: object[] = [
      { groupPath: 'Country', headerTemplate: '<div>{{:Country}}</div>',
headerPosition: 'Center', border: { color: 'black', width: 0.5 } },
      { groupPath:
'JobDescription', headerTemplate: '<div>{{:JobDescription}}</div>',
headerPosition: 'Center', border: { color: 'black', width: 0.5 } },
      { groupPath: 'JobGroup', headerTemplate: '<div>{{:JobGroup}}</div>',
headerPosition: 'Far', border: { color: 'black', width: 0.5 } },
    ]
    public palette: object = ["#f44336", "#29b6f6", "#ab47bc", "#ffc107",
"#5c6bc0", "#009688"];
  }

```

MAIN.TS

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

Color mapping in Angular Treemap component

Color mapping is used to customize the color for each group or item based on the specified types. The following options are available to customize the group and leaf items in the TreeMap.

Range color mapping

Range color mapping is used to apply color to the items by giving specific ranges in the DataSource, and it should be specifying the data source properties to the [rangeColorValuePath](#). The color mapping ranges to be specified in the [from](#) and [to](#) properties of the [colorMapping](#).

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' rangeColorValuePath='count'
[leafItemSettings]='leafItemSettings'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [{ fruit:'Apple', count:5000 },
    { fruit:'Mango', count:3000 },
    { fruit:'Orange', count:2300 },
    { fruit:'Banana', count:500 },
    { fruit:'Grape', count:4300 },
    { fruit:'Papaya', count:1200 },
    { fruit:'Melon', count:4500 }
  ];
  public leafItemSettings: object = {
    labelPath: 'fruit',
    colorMapping:[
      {
        from:500,
        to:3000,
        color:'orange'
      },
      {
        from:3000,
```

```

        to:5000,
        color:'green'
    }
  ]
};
}

```

MAIN.TS

```

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

```

Equal color mapping

Equal color mapping is used to fill colors to each item by specifying equal value present in the data source, that can be specified in the [equalColorValuePath](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' equalColorValuePath='Brand'
[leafItemSettings]='leafItemSettings'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    { Car:'Mustang', Brand:'Ford', count:232 },
    { Car:'EcoSport', Brand:'Ford', count:121 },
    { Car:'Swift', Brand:'Maruti', count:143 },
    { Car:'Baleno', Brand:'Maruti', count:454 },
    { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
    { Car:'A3 Cabriolet', Brand:'Audi', count:123 },
    { Car:'RS7 Sportback', Brand:'Audi', count:523 }
  ];
  public leafItemSettings: object = {
    labelPath: 'Car',
    colorMapping:[
      {
        value:'Ford',
        color:'green'
      },
    ],
  }
}

```

```

        value: 'Audi',
        color: 'red'
      },
      {
        value: 'Maruti',
        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));

```

Desaturation color mapping

Desaturation color mapping is used to apply colors to the items based on the [minOpacity](#) and [maxOpacity](#) properties in the [colorMapping](#).

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap';
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' rangeColorValuePath='count'
[leafItemSettings]='leafItemSettings'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    { fruit: 'Apple', count: 5000 },
    { fruit: 'Mango', count: 3000 },
    { fruit: 'Orange', count: 2300 },
    { fruit: 'Banana', count: 500 },
    { fruit: 'Grape', count: 4300 },
    { fruit: 'Papaya', count: 1200 },
    { fruit: 'Melon', count: 4500 }
  ];
  public leafItemSettings: object = {
    labelPath: 'fruit',
    colorMapping: [
      {

```



```

        from:500,
        to:3000,
        minOpacity:0.2,
        maxOpacity:0.5,
        color:'orange'
    },
    {
        from:3000,
        to:5000,
        minOpacity:0.5,
        maxOpacity:0.8,
        color:'green'
    }
]
};
}

```

MAIN.TS

```

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

```

Palette color mapping

The palette color mapping is used to fill the color to each group or leaf item by given colors in the [palette](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap';
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' [palette]='palette'
[leafItemSettings]='leafItemSettings'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    { Car:'Mustang', Brand:'Ford', count:232 },
    { Car:'EcoSport', Brand:'Ford', count:121 },
    { Car:'Swift', Brand:'Maruti', count:143 },
    { Car:'Baleno', Brand:'Maruti', count:454 },
    { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
    { Car:'A3 Cabriolet', Brand:'Audi',count:123 },

```

```

        { car:'RS7 Sportback', Brand:'Audi', count:523 }
    ];
    public leafItemSettings: object = {
        labelPath: 'Car'
    };
    public palette: object= ['red','green'];
}

```

MAIN.TS

```

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

```

Desaturation with multiple colors

Multiple colors are used as gradient effect to specific shapes based on the ranges in datasource. By using [color](#) property, you can set n number of colors.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' rangeColorValuePath='count'
[leafItemSettings]='leafItemSettings' [legendSettings] = 'legendSettings'>
    </ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    { fruit:'Apple', count:5000 },
    { fruit:'Mango', count:3000 },
    { fruit:'Orange', count:2300 },
    { fruit:'Banana', count:500 },
    { fruit:'Grape', count:4300 },
    { fruit:'Papaya', count:1200 },
    { fruit:'Melon', count:4500 }
  ];
  public legendSettings:object ={
    visible:true
  }
  public leafItemSettings: object = {
    labelPath: 'fruit',
    colorMapping:[
      {

```

```

        from:500,
        to:2500,
        color:['orange','pink']
    },
    {
        from:3000,
        to:5000,
        color:['green','red','blue']
    }
]
};
}

```

MAIN.TS

```

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

```

Color for items excluded from color mapping

Get the excluded ranges from data source using the color mapping and apply the specific color to those items, without specifying the [from](#) and [to](#) properties.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' rangeColorValuePath='count'
[leafItemSettings]='leafItemSettings'>
  </ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    { fruit:'Apple', count:5000 },
    { fruit:'Mango', count:3000 },
    { fruit:'Orange', count:2300 },
    { fruit:'Banana', count:500 },
    { fruit:'Grape', count:4300 },
    { fruit:'Papaya', count:1200 },
    { fruit:'Melon', count:4500 }
  ];
  public leafItemSettings: object = {
    labelPath: 'fruit',

```

```
        colorMapping:[
            {
                from:500,
                to:2500,
                color:'orange'
            },
            {
                from:3000,
                to:4000,
                color:'green'
            },
            {
                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));
```

Bind the colors to the items from data source

To set the color for each item from the data source, bind the data source property to the [colorValuePath](#).

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap';
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' colorValuePath='color'
[leafItemSettings]='leafItemSettings'>
    </ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    { fruit:'Apple', count:5000, color: 'red'},
    { fruit:'Mango', count:3000, color:'blue'},
    { fruit:'Orange', count:2300, color:'green' },
    { fruit:'Banana', count:500, color:'yellow' },
    { fruit:'Grape', count:4300, color:'orange' },
```

```

    { fruit: 'Papaya', count: 1200, color: 'pink' },
    { fruit: 'Melon', count: 4500, color: 'violet' }
  ];
  public leafItemSettings: object = {
    labelPath: 'fruit'
  };
}

```

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 Treemap component

Data Labels are used to identify the name of items or groups in the TreeMap component. Data Labels will be shown by specifying the data source properties in the [labelPath](#) of the [leafItemSettings](#).

Format

Customize the labels for each item using the [labelFormat](#) property in the [leafItemSettings](#).

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap';
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block; height=
'350px' [dataSource]='data' weightValuePath='count' [leafItemSettings]='leafItemSettings'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    { Car: 'Mustang', Brand: 'Ford', count: 232 },
    { Car: 'EcoSport', Brand: 'Ford', count: 121 },
    { Car: 'Swift', Brand: 'Maruti', count: 143 },
    { Car: 'Baleno', Brand: 'Maruti', count: 454 },
    { Car: 'Vitara Brezza', Brand: 'Maruti', count: 545 },
    { Car: 'A3 Cabriolet', Brand: 'Audi', count: 123 },
    { Car: 'RS7 Sportback', Brand: 'Audi', count: 523 }
  ];
  public leafItemSettings: object = {
    labelPath: 'Car',
    format: `${Car}-${Brand}`
  };
}

```

```
}

```

MAIN.TS

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

```

Template

The template supports customizing labels of each leaf node using the [labelTemplate](#) property. It uses Essential JS2 template engine to render elements and the position of templates can be customize using the [templatePosition](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;
height='350px' [dataSource]='data' weightValuePath='count'
[leafItemSettings]='leafItemSettings'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    { Car: 'Mustang', Brand: 'Ford', count: 232 },
    { Car: 'EcoSport', Brand: 'Ford', count: 121 },
    { Car: 'Swift', Brand: 'Maruti', count: 143 },
    { Car: 'Baleno', Brand: 'Maruti', count: 454 },
    { Car: 'Vitara Brezza', Brand: 'Maruti', count: 545 },
    { Car: 'A3 Cabriolet', Brand: 'Audi', count: 123 },
    { Car: 'RS7 Sportback', Brand: 'Audi', count: 523 }
  ];
  public leafItemSettings: object = {
    labelPath: 'Car',
    labelTemplate: '<div>{{:Car}}-{{:Brand}}</div>',
    templatePosition: 'Center'
  };
}
```

MAIN.TS

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

```

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

InterSectAction

When the label size in each item exceeds the actual size, use the [interSectAction](#) property in the [leafItemSettings](#) to customise the labels.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
height='350px' [dataSource]='data' weightValuePath='count'
[leafItemSettings]='leafItemSettings'>
  </ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    { Car: 'Mustang', Brand: 'Ford', count: 232 },
    { Car: 'EcoSport', Brand: 'Ford', count: 121 },
    { Car: 'Swift', Brand: 'Maruti', count: 143 },
    { Car: 'Baleno', Brand: 'Maruti', count: 454 },
    { Car: 'Vitara Brezza', Brand: 'Maruti', count: 545 },
    { Car: 'A3 Cabriolet', Brand: 'Audi', count: 123 },
    { Car: 'RS7 Sportback', Brand: 'Audi', count: 523 }
  ];
  public leafItemSettings: object = {
    labelPath: 'Car',
    format: '${Car}-${Brand}',
    interSectAction: 'WrapByWord'
  };
}
```

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 Treemap component

Legend is used to provide valuable information for interpreting what the TreeMap displays. The legends can be represented in various colors, shapes or other identifiers based on the data.

Position and alignment

Legend position is used to place legend in various positions. Based on the legend position, the legend item will be aligned. For example, if the position is top or bottom, the legend items are placed by rows. If the position is left or right, the legend items are placed by columns.

The following options are available to customize the legend position:

- Top
- Bottom
- Left
- Right
- Float

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' [legendSettings]='legendSettings'
[leafItemSettings]='leafItemSettings' [rangeColorValuePath]=
'rangeColorValuePath'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [{ fruit:'Apple', count:5000 },
    { fruit:'Mango', count:3000 },
    { fruit:'Orange', count:2300 },
    { fruit:'Banana', count:500 },
    { fruit:'Grape', count:4300 },
    { fruit:'Papaya', count:1200 },
    { fruit:'Melon', count:4500 }
  ];
  public rangeColorValuePath: object | string = 'count';
  public legendSettings: object= {
    visible: true,
    position: 'Top'
  };
  public leafItemSettings:object= {
    labelPath: 'fruit',
    colorMapping:[
      {
        from:500,
        to:3000,
        color:'orange'
      },
    ],
  },
}
```



```

    {
      from: 3000,
      to: 5000,
      color: 'green'
    }
  ]
}

```

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 is used to align the legend items in specific location. The following options are available to customize the legend alignment:

- Near
- Center
- Far

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap';
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' [legendSettings]='legendSettings'
[leafItemSettings]='leafItemSettings' [rangeColorValuePath]=
'rangeColorValuePath'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [{ fruit: 'Apple', count: 5000 },
    { fruit: 'Mango', count: 3000 },
    { fruit: 'Orange', count: 2300 },
    { fruit: 'Banana', count: 500 },
    { fruit: 'Grape', count: 4300 },
    { fruit: 'Papaya', count: 1200 },
    { fruit: 'Melon', count: 4500 }
  ];
  public rangeColorValuePath: object | string = 'count';
  public legendSettings: object = {

```

```

        visible: true,
        alignment: 'Far',
    };
    public leafItemSettings: object = {
        labelPath: 'fruit',
        colorMapping: [
            {
                from: 500,
                to: 3000,
                color: 'orange'
            },
            {
                from: 3000,
                to: 5000,
                color: 'green'
            }
        ]
    }
}

```

MAIN.TS

```

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

```

Legend mode

The TreeMap control supports two different types of legend rendering modes such as **Default** and **Interactive**.

<!-- markdownlint-disable MD036 -->

Default mode

In default mode, the legends have symbols with legend labels that are used to identify the items in the TreeMap.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' [legendSettings]='legendSettings'
[leafItemSettings]='leafItemSettings' equalColorValuePath='Brand'>
</ejs-treemap>`

```

```

    })
    export class AppComponent {
      public data: object[] = [{ Car:'Mustang', Brand:'Ford', count:232 },
                                { Car:'EcoSport', Brand:'Ford', count:121 },
                                { Car:'Swift', Brand:'Maruti', count:143 },
                                { Car:'Baleno', Brand:'Maruti', count:454 },
                                { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
                                { Car:'A3 Cabriolet', Brand:'Audi', count:123 },
                                { Car:'RS7 Sportback', Brand:'Audi', count:523 }
                                ];
      public legendSettings: object= {
        visible: true,
        position:'Top',
        border:{color:'black',width:2}
      };
      public leafItemSettings:object= {
        labelPath: 'Car',
        colorMapping:[
          {
            value:'Ford',
            color:'green'
          },
          {
            value:'Audi',
            color:'red'
          },
          {
            value:'Maruti',
            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));

```

<!-- markdownlint-disable MD036 -->

Interactive mode

The legends can be made interactive with an arrow mark that indicates exact range color in the legend when the mouse hovers on the TreeMap item. Enable this option by setting the [mode](#) property in the [legendSettings](#) to **Interactive**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({

```

```

imports: [
    TreeMapModule, TreeMapAllModule
],
providers: [TreeMapLegendService, TreeMapTooltipService],
standalone: true,
selector: 'app-container',
template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' [legendSettings]='legendSettings'
[leafItemSettings]='leafItemSettings' equalColorValuePath='Brand'>
</ejs-treemap>`
})
export class AppComponent {
    public data: object[] = [{ Car:'Mustang', Brand:'Ford', count:232 },
                            { Car:'EcoSport', Brand:'Ford', count:121 },
                            { Car:'Swift', Brand:'Maruti', count:143 },
                            { Car:'Baleno', Brand:'Maruti', count:454 },
                            { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
                            { Car:'A3 Cabriolet', Brand:'Audi', count:123 },
                            { Car:'RS7 Sportback', Brand:'Audi', count:523 }
    ];
    public legendSettings: object= {
        visible: true,
        mode:'Interactive',
        position:'Top',
        border:{color:'black',width:2}
    };
    public leafItemSettings:object= {
        labelPath: 'Car',
        colorMapping:[
            {
                value:'Ford',
                color:'green'
            },
            {
                value:'Audi',
                color:'red'
            },
            {
                value:'Maruti',
                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));

```

Legend size

Customize the legend size by modifying the [height](#) and [width](#) properties in the [legendSettings](#). It accepts values in both percentage and pixel.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' [legendSettings]='legendSettings'
[leafItemSettings]='leafItemSettings' equalColorValuePath='Brand'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [{ Car:'Mustang', Brand:'Ford', count:232 },
    { Car:'EcoSport', Brand:'Ford', count:121 },
    { Car:'Swift', Brand:'Maruti', count:143 },
    { Car:'Baleno', Brand:'Maruti', count:454 },
    { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
    { Car:'A3 Cabriolet', Brand:'Audi',count:123 },
    { car:'RS7 Sportback', Brand:'Audi', count:523 }
  ];
  public legendSettings: object= {
    visible: true,
    height: '50px',
    width: '200px',
    position: 'Top',
    border:{color: 'black',width:2}
  };
  public leafItemSettings:object= {
    labelPath: 'Car',
    colorMapping:[
      {
        value: 'Ford',
        color: 'green'
      },
      {
        value: 'Audi',
        color: 'red'
      },
      {
        value: 'Maruti',
        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));
```

Paging support

TreeMap support legend paging, if the legend items cannot be placed within the provided [height](#) and [width](#) of the legend.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' [legendSettings]='legendSettings'
[leafItemSettings]='leafItemSettings' equalColorValuePath='Brand'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [{ Car:'Mustang', Brand:'Ford', count:232 },
    { Car:'EcoSport', Brand:'Ford', count:121 },
    { Car:'Swift', Brand:'Maruti', count:143 },
    { Car:'Baleno', Brand:'Maruti', count:454 },
    { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
    { Car:'A3 Cabriolet', Brand:'Audi', count:123 },
    { Car:'RS7 Sportback', Brand:'Audi', count:523 }
  ];
  public legendSettings: object= {
    visible: true,
    height:'50px',
    width:'100px',
    border:{color:'black',width:2}
  };
  public leafItemSettings:object= {
    labelPath: 'Car',
    colorMapping:[
      {
        value:'Ford',
        color:'green'
      },
      {
        value:'Audi',
        color:'red'
      },
      {
        value:'Maruti',
```

```

        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));

```

Legend for items excluded from color mapping

Based on the mapping ranges in the data source, get the excluded ranges from the color mapping, and show the legend with the excluded range values that are bound to the specific legend.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap';
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' [legendSettings]='legendSettings'
[leafItemSettings]='leafItemSettings' rangeColorValuePath='count'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [{ fruit: 'Apple', count: 5000 },
    { fruit: 'Mango', count: 3000 },
    { fruit: 'Orange', count: 2300 },
    { fruit: 'Banana', count: 500 },
    { fruit: 'Grape', count: 4300 },
    { fruit: 'Papaya', count: 1200 },
    { fruit: 'Melon', count: 4500 }
  ];
  public legendSettings: object = {
    visible: true
  };
  public leafItemSettings: object = {
    labelPath: 'fruit',
    colorMapping: [
      {
        from: 500,
        to: 2500,
        color: 'orange'
      }
    ]
  };
}

```

```

    },
    {
      from: 3000,
      to: 4000,
      color: 'green'
    },
    {
      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));

```

Hide desired legend items

To enable or disable the desired legend item for each color mapping, set the [showLegend](#) property to **true** in the [colorMapping](#).

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap';
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' [legendSettings]='legendSettings'
[leafItemSettings]='leafItemSettings' rangeColorValuePath='count'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [{ fruit: 'Apple', count: 5000 },
    { fruit: 'Mango', count: 3000 },
    { fruit: 'Orange', count: 2300 },
    { fruit: 'Banana', count: 500 },
    { fruit: 'Grape', count: 4300 },
    { fruit: 'Papaya', count: 1200 },
    { fruit: 'Melon', count: 4500 }
  ];
  public legendSettings: object = {
    visible: true
  };
};

```



```

public leafItemSettings:object= {
  labelPath: 'fruit',
  colorMapping:[
    {
      from:500,
      to:2500,
      color:'orange',
      showLegend: true
    },
    {
      from:3000,
      to:5000,
      color:'green',
      showLegend: false
    },
    {
      color:'red',
      showLegend: true
    }
  ]
}

```

MAIN.TS

```

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

```

Hide legend items based data source value

To enable or disable the legend visibility for each item through the data source, bind the appropriate data source field name to [showLegendPath](#) property in the [legendSettings](#).

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' [legendSettings]='legendSettings'
[leafItemSettings]='leafItemSettings' colorValuePath='color'>
</ejs-treemap>`
})
export class AppComponent {

```

```

    public data: object[] = [{ fruit: 'Apple', count: 5000, visibility: true,
    color: 'red' },
                                { fruit: 'Mango', count: 3000, visibility: false,
    color: 'blue' },
                                { fruit: 'Orange', count: 2300, visibility: true,
    color: 'green' },
                                { fruit: 'Banana', count: 500, visibility: false,
    color: 'yellow' },
                                { fruit: 'Grape', count: 4300, visibility: true,
    color: 'blue' },
                                { fruit: 'Papaya', count: 1200, visibility: false,
    color: 'black' },
                                { fruit: 'Melon', count: 4500, visibility: true,
    color: 'skyblue' }
    ];
    public legendSettings: object = {
        visible: true,
        showLegendPath: 'visibility'
    };
    public leafItemSettings: object = {
        labelPath: 'fruit',
    }
}

```

MAIN.TS

```

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

```

Bind legend item text from data source

To show the legend item text from the data source, bind the property name from data source to the [valuePath](#) property in the [legendSettings](#).

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap';
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' [legendSettings]='legendSettings'
[leafItemSettings]='leafItemSettings' colorValuePath='color'>
</ejs-treemap>`
})
export class AppComponent {

```

```

    public data: object[] = [{ fruit:'Apple', count:5000, visibility: true,
    color:'red' },
                                { fruit:'Mango', count:3000, visibility: false,
    color:'blue' },
                                { fruit:'Orange', count:2300, visibility: true,
    color:'green' },
                                { fruit:'Banana', count:500, visibility: false,
    color:'yellow'},
                                { fruit:'Grape', count:4300, visibility: true,
    color:'blue' },
                                { fruit:'Papaya', count:1200, visibility: false,
    color:'black' },
                                { fruit:'Melon', count:4500, visibility: true,
    color:'skyblue' }
    ];
    public legendSettings: object= {
        visible: true,
        valuePath: 'fruit'
    };
    public leafItemSettings:object= {
        labelPath: 'fruit',
    };
}

```

MAIN.TS

```

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

```

Hide duplicate legend items

To enable or disable the duplicate legend items, set the [removeDuplicateLegend](#) property to **true** in the [legendSettings](#).

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' [legendSettings]='legendSettings'
[leafItemSettings]='leafItemSettings' colorValuePath='color'>
    </ejs-treemap>`
})
export class AppComponent {

```

```

    public data: object[] = [{ fruit:'Apple', count:5000, visibility: true,
    color:'red' },
                                { fruit:'Mango', count:3000, visibility: false,
    color:'blue' },
                                { fruit:'Orange', count:2300, visibility: true,
    color:'green' },
                                { fruit:'Banana', count:500, visibility: false,
    color:'yellow' },
                                { fruit:'Grape', count:4300, visibility: true,
    color:'blue' },
                                { fruit:'Papaya', count:1200, visibility: false,
    color:'black' },
                                { fruit:'Melon', count:4500, visibility: true,
    color:'skyblue' }
    ];
    public legendSettings: object= {
        visible: true,
        valuePath:'fruit',
        removeDuplicateLegend: true
    };
    public leafItemSettings:object= {
        labelPath: 'fruit',
    };
}

```

MAIN.TS

```

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

```

Legend Responsiveness

Use a responsive legend that switches positions between the right and the bottom based on the available height and width. To enable the responsive legend, set the [position](#) property to **Auto** in the [legendSettings](#) and the legend position is changed based on the available height and width.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;' [palette]
='palette' [dataSource]='data' weightValuePath='count'
[legendSettings] ='legendSettings'>
</ejs-treemap>`
})

```

```

    })
    export class AppComponent {
        public data: object[] = [{ fruit:'Apple', count:5000 },
                                { fruit:'Mango', count:3000 },
                                { fruit:'Orange', count:2300 },
                                { fruit:'Banana', count:500 },
                                { fruit:'Grape', count:4300 },
                                { fruit:'Papaya', count:1200 },
                                { fruit:'Melon', count:4500 }
                                ];
        public leafItemSettings:object= {
            labelPath: 'fruit'
        };
        public palette:object =['#71B081', '#5A9A77', '#498770', '#39776C',
                                '#266665', '#124F5E'];
        public legendSettings: object= {
            visible:true,
            position:'Auto'
        };
    }
}

```

MAIN.TS

```

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

```

Drilldown in Angular Treemap component

The TreeMap supports drill-down to expose the hierarchy, achieved by clicking a node. If an item is clicked in the TreeMap, it will be moved to the next level or sub level hierarchy and returned back to the previous level by clicking the node.

Perform drill-down action

The TreeMap items can be drilled by setting the [enableDrillDown](#) property to **true**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='EmployeesCount' enableDrillDown= true
[levels]='levels' [palette]='palette'>
</ejs-treemap>`
})

```

```

export class AppComponent {
  public data: object[] = [
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
      JobGroup: 'Executive', EmployeesCount: 20 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
      JobGroup: 'Analyst', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
      EmployeesCount: 40 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
      EmployeesCount: 80 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
      JobGroup: 'Testers', EmployeesCount: 100 },
    { Category: 'Employees', Country: 'India', JobDescription: 'HR
      Executives', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
      EmployeesCount: 40 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
      JobGroup: 'Executive', EmployeesCount: 50 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
      JobGroup: 'Analyst', EmployeesCount: 60 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Marketing',
      EmployeesCount: 70 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Technical',
      JobGroup: 'Testers', EmployeesCount: 80 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
      'Management', EmployeesCount: 10 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
      EmployeesCount: 20 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
      JobGroup: 'Testers', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
      EmployeesCount: 50 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
      EmployeesCount: 60 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
      JobGroup: 'Testers', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
      EmployeesCount: 100 }
  ];
  public palette: object= ["#f44336", "#29b6f6", "#ab47bc", "#ffc107",
    "#5c6bc0", "#009688"];
  public levels: object = [
    { groupPath: 'Country', border: { color: 'black', width: 0.5 } },
    { groupPath: 'JobDescription', border: { color: 'black', width: 0.5 } }
  ],
  { groupPath: 'JobGroup', border: { color: 'black', width: 0.5 } },
]
}

```

MAIN.TS

```

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

```

On-demand data loading

All the child items are rendered during the normal drill-down process, and visible at the initial rendering of the TreeMap. But on-demand data loading, it will not render child items at initial rendering, and child nodes will be rendered during the drill-down process by setting the [drillDownView](#) property to **true**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='EmployeesCount' enableDrillDown= true
drillDownView= true
[levels]='levels' [palette]='palette'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 20 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
EmployeesCount: 80 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 100 },
    { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
EmployeesCount: 40 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Executive', EmployeesCount: 50 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
JobGroup: 'Analyst', EmployeesCount: 60 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Marketing',
EmployeesCount: 70 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 80 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
EmployeesCount: 20 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
JobGroup: 'Testers', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
EmployeesCount: 50 },
```

```

    { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
      EmployeesCount: 60 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
      JobGroup: 'Testers', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
      EmployeesCount: 100 }
  ];
  public palette: object= ["#f44336", "#29b6f6", "#ab47bc", "#ffc107",
    "#5c6bc0", "#009688"];
  public levels: object = [
    { groupPath: 'Country', border: { color: 'black', width: 0.5 } },
    { groupPath: 'JobDescription', border: { color: 'black', width: 0.5 } },
  ],
  { groupPath: 'JobGroup', border: { color: 'black', width: 0.5 } },
  ]
}

```

MAIN.TS

```

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

```

Breadcrumb support

TreeMap items are drilled, up to any level of parent using breadcrumb navigation and the level from root parent to current level is displayed at the top of item layout. It can be enabled by using the [enableBreadcrumb](#) property to **true** and customize the breadcrumb connector using the [breadcrumbConnector](#) property. By default, **-(hyphen)** is the connector.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
  TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
    [dataSource]='data' weightValuePath='EmployeesCount' enableDrillDown= true
    enableBreadcrumb= true breadcrumbConnector= ' -> '
    [levels]='levels' [palette]='palette'>
  </ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
      JobGroup: 'Executive', EmployeesCount: 20 },
  ],

```



```

    { Category: 'Employees', Country: 'USA', JobDescription: 'Sales',
      JobGroup: 'Analyst', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Marketing',
      EmployeesCount: 40 },
    { Category: 'Employees', Country: 'USA', JobDescription: 'Management',
      EmployeesCount: 80 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Technical',
      JobGroup: 'Testers', EmployeesCount: 100 },
    { Category: 'Employees', Country: 'India', JobDescription: 'HR
      Executives', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'India', JobDescription: 'Accounts',
      EmployeesCount: 40 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
      JobGroup: 'Executive', EmployeesCount: 50 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Sales',
      JobGroup: 'Analyst', EmployeesCount: 60 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Marketing',
      EmployeesCount: 70 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Technical',
      JobGroup: 'Testers', EmployeesCount: 80 },
    { Category: 'Employees', Country: 'Germany', JobDescription:
      'Management', EmployeesCount: 10 },
    { Category: 'Employees', Country: 'Germany', JobDescription: 'Accounts',
      EmployeesCount: 20 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Technical',
      JobGroup: 'Testers', EmployeesCount: 30 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'HR Executives',
      EmployeesCount: 50 },
    { Category: 'Employees', Country: 'UK', JobDescription: 'Accounts',
      EmployeesCount: 60 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Technical',
      JobGroup: 'Testers', EmployeesCount: 70 },
    { Category: 'Employees', Country: 'France', JobDescription: 'Marketing',
      EmployeesCount: 100 }
  ];
  public palette: object= ["#f44336", "#29b6f6", "#ab47bc", "#ffc107",
    "#5c6bc0", "#009688"];
  public levels: object = [
    { groupPath: 'Country', border: { color: 'black', width: 0.5 } },
    { groupPath: 'JobDescription', border: { color: 'black', width: 0.5 } }
  ],
  { groupPath: 'JobGroup', border: { color: 'black', width: 0.5 } },
  ]
}

```

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 Treemap component

Tooltip is used to display details about the items in the TreeMap. When space constraints prevent us from displaying the information using Data Labels, the tooltip comes in handy.

Default tooltip

The tooltip is not visible by default, to make it visible, set the [visible](#) property in the [tooltipSettings](#) to **true**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count'
[leafItemSettings]='leafItemSettings'
[tooltipSettings]='tooltipSettings'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [{ fruit:'Apple', count:5000 },
    { fruit:'Mango', count:3000 },
    { fruit:'Orange', count:2300 },
    { fruit:'Banana', count:500 },
    { fruit:'Grape', count:4300 },
    { fruit:'Papaya', count:1200 },
    { fruit:'Melon', count:4500 }
  ];
  public leafItemSettings: object = {
    labelPath: 'fruit'
  };
  public tooltipSettings: object = {
    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));
```

Format tooltip

The tooltip content is displayed by default based on the [weightValuePath](#). In addition, to show more information in the tooltip, use the [format](#) property and define field from the data source as ``${datafield}``.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count'
[leafItemSettings]='leafItemSettings'
[tooltipSettings]='tooltipSettings'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [{ fruit:'Apple', count:5000 },
    { fruit:'Mango', count:3000 },
    { fruit:'Orange', count:2300 },
    { fruit:'Banana', count:500 },
    { fruit:'Grape', count:4300 },
    { fruit:'Papaya', count:1200 },
    { fruit:'Melon', count:4500 }
  ];
  public leafItemSettings: object = {
    labelPath: 'fruit'
  };
  public tooltipSettings: object = {
    visible: true,
    format: 'Name:${fruit} - TotalCount:${count}'
  };
}

```

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

Tooltip can be rendered as a custom component using the [template](#) property in the [tooltipSettings](#) which accepts one or more UI elements as an input, that can be rendered as a part of the tooltip rendering. You can use `${datafield}` as placeholder in HTML element to display the values from data source.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'

```

```
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count'
[leafItemSettings]='leafItemSettings'
[tooltipSettings]='tooltipSettings'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [{ fruit:'Apple', count:5000 },
    { fruit:'Mango', count:3000 },
    { fruit:'Orange', count:2300 },
    { fruit:'Banana', count:500 },
    { fruit:'Grape', count:4300 },
    { fruit:'Papaya', count:1200 },
    { fruit:'Melon', count:4500 }
  ];
  public leafItemSettings: object = {
    labelPath: 'fruit'
  };
  public tooltipSettings: object = {
    visible: true,
    template: '<div><p>Name: ${fruit}</p><p>Total Count:
${count}</p></div>'
  };
}
```

MAIN.TS

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

Selection and highlight in Angular Treemap component

Selection

Selection is used to select a particular group or item to differentiate from other items. Each item or each group can be selected and deselected while interacting with the items. The corresponding Treemap items are also selected while tapping a specific legend item, and vice versa.

The [fill](#) property is used to change the selected item color. The [color](#) and the [width](#) properties are used to customize the selected item border, and the selection is enabled by using the [enable](#) property to **true** in the [selectionSettings](#).

APP.COMPONENT.TS

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

```

import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='sales'
[leafItemSettings]='leafItemSettings' [selectionSettings]=
'selectionSettings' [levels]='levels'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    { dataType: "Import", type: "Animal products", product: "2010", sales:
20839332874 },
    { dataType: "Import", type: "Animal products", product: "2011", sales:
23098635589 },
    { dataType: "Import", type: "Chemical products", product: "2010", sales:
141637951510 },
    { dataType: "Import", type: "Chemical products", product: "2011", sales:
161550338209 },
    { dataType: "Import", type: "Base metals", product: "2010", sales:
86079439944 },
    { dataType: "Import", type: "Base metals", product: "2011", sales:
103821671535 },
    { dataType: "Import", type: "Textile articles", product: "2010",
sales: 97126140830 },
    { dataType: "Import", type: "Textile articles", product: "2011",
sales: 104980750811 },
    { dataType: "Export", type: "Animal products", product: "2010", sales:
15845503378 },
    { dataType: "Export", type: "Animal products", product: "2011", sales:
20650111620 },
    { dataType: "Export", type: "Chemical products", product: "2010", sales:
136100054087 },
    { dataType: "Export", type: "Chemical products", product: "2011", sales:
146341672411 },
    { dataType: "Export", type: "Base metals", product: "2010", sales:
59060592813 },
    { dataType: "Export", type: "Base metals", product: "2011", sales:
71785882641 },
    { dataType: "Export", type: "Textile articles", product: "2010",
sales: 20982380561 },
    { dataType: "Export", type: "Textile articles", product: "2011",
sales: 26016143783 }
  ];
  public levels: object[] = [
    { groupPath: 'dataType', fill: '#c5e2f7', headerStyle: { size:
'16px' }, headerAlignment: 'Center', groupGap: 5 },
    { groupPath: 'product', fill: '#a4d1f2', headerAlignment:
'Center' , groupGap: 2 }
  ];
}

```

```

    public leafItemSettings: object= {
        labelPath: 'type',
        fill: '#8ebfe2',
        labelPosition: 'Center'
    };
    public selectionSettings: object= {
        enable: true,
        fill: '#58a0d3',
        border: { width: 0.3, color: 'black' },
        opacity: '1'
    };
}

```

MAIN.TS

```

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

```

Highlight

Highlight is used to highlight an item or group from other items or groups. Each item or each group can be highlighted by hovering the mouse over the items. The corresponding Treemap items are also be highlighted while hovering over a specific legend item, and vice versa.

The [fill](#) property is used to change the highlighted item color. The [color](#) and the [width](#) properties are used to customize the highlighted item border, and the highlight is enabled by setting the [enable](#) property to **true** in the [highlightSettings](#).

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='sales'
[leafItemSettings]='leafItemSettings' [highlightSettings]=
'highlightSettings' [levels]='levels'>
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [
    { dataType: "Import", type: "Animal products", product: "2010", sales:
20839332874 },
    { dataType: "Import", type: "Animal products", product: "2011", sales:
23098635589 },

```

```

    { dataType: "Import", type: "Chemical products", product: "2010", sales:
141637951510 },
    { dataType: "Import", type: "Chemical products", product: "2011", sales:
161550338209 },
    { dataType: "Import", type: "Base metals", product: "2010", sales:
86079439944 },
    { dataType: "Import", type: "Base metals", product: "2011", sales:
103821671535 },
    { dataType: "Import", type: "Textile articles", product: "2010",
sales: 97126140830 },
    { dataType: "Import", type: "Textile articles", product: "2011",
sales: 104980750811 },
    { dataType: "Export", type: "Animal products", product: "2010", sales:
15845503378 },
    { dataType: "Export", type: "Animal products", product: "2011", sales:
20650111620 },
    { dataType: "Export", type: "Chemical products", product: "2010", sales:
136100054087 },
    { dataType: "Export", type: "Chemical products", product: "2011", sales:
146341672411 },
    { dataType: "Export", type: "Base metals", product: "2010", sales:
59060592813 },
    { dataType: "Export", type: "Base metals", product: "2011", sales:
71785882641 },
    { dataType: "Export", type: "Textile articles", product: "2010",
sales: 20982380561 },
    { dataType: "Export", type: "Textile articles", product: "2011",
sales: 26016143783 }
  ];
  public levels: object[] = [
    { groupPath: 'dataType', fill: '#c5e2f7', headerStyle: { size:
'16px' }, headerAlignment: 'Center', groupGap: 5 },
    { groupPath: 'product', fill: '#a4d1f2', headerAlignment:
'Center' , groupGap: 2 }
  ];
  public leafItemSettings: object = {
    labelPath: 'type',
    fill: '#8ebfe2',
    labelPosition: 'Center'
  };
  public highlightSettings: object = {
    enable: true,
    fill: '#71b0dd',
    border: { width: 0.3, color: 'black' },
    opacity: '1'
  };
}

```

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 Treemap 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 treemap 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 { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { TreeMap } from '@syncfusion/ej2-treemap';
import { Component, ViewChild } from '@angular/core';
import { PrintService } from '@syncfusion/ej2-angular-treemap';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' #treemap [allowPrint]=true
style='display: block;' [dataSource]='data' weightValuePath='GDP'
[leafItemSettings]='leafItemSettings'>
</ejs-treemap> <div>
<button id="togglebtn2" (click)='print()'>Print</button>
</div>`,
  providers: [PrintService]
})
export class AppComponent {
  @ViewChild('treemap')
  public treemap?: TreeMap;
  public data: object[] = [
    {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},
    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
  ];
  public leafItemSettings: object = {
    labelPath: 'State',
    labelFormat: '${State}<br>${GDP} Trillion<br>(${percentage} %)',
    labelStyle: {
      color: '#000000'
    },
    border: {
      color: '#000000',
      width: 0.5
    },
  },
};
```



```

    public print() {
        this.treemap!.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

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 treemap can be exported as an image using the `export` method. The method requires two parameters: image type and file name. The treemap 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 { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap';
import { TreeMap } from '@syncfusion/ej2-treemap';
import { Component, ViewChild } from '@angular/core';
import { ImageExportService } from '@syncfusion/ej2-angular-treemap';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' #treemap [allowImageExport]=true
style='display: block;' [dataSource]='data' weightValuePath='GDP'
[leafItemSettings]='leafItemSettings'>
</ejs-treemap> <div>
<button id="togglebtn2" (click)='export()'>Export</button>
</div>`,
  providers: [ImageExportService]
})
export class AppComponent {
  @ViewChild('treemap')
  public treemap?: TreeMap;
  public data: object[] = [
    {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},

```

```

    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
  ];
  public leafItemSettings: object = {
    labelPath: 'State',
    labelFormat: '${State}<br>${GDP} Trillion<br>(${percentage} %)',
    labelStyle: {
      color: '#000000'
    },
    border: {
      color: '#000000',
      width: 0.5
    },
  };
  public export() {
    this.treemap!.export('PNG', 'TreeMap');
  };
}

```

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 treemap 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 { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap';
import { TreeMap } from '@syncfusion/ej2-treemap';
import { Component, ViewChild } from '@angular/core';
import { ImageExportService } from '@syncfusion/ej2-angular-treemap';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' #treemap [allowImageExport]=true
style='display: block;' [dataSource]='data' weightValuePath='GDP'

```

```

[leafItemSettings]='leafItemSettings'>
</ejs-treemap> <div>
<button id="togglebtn2" (click)='export()'>Export</button>
</div>`,
providers: [ImageExportService]
})
export class AppComponent {
  @ViewChild('treemap')
  public treemap?: TreeMap;
  public data: object[] = [
    {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},
    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
  ];
  public leafItemSettings: object = {
    labelPath: 'State',
    labelFormat: '${State}<br>${GDP} Trillion<br>(${percentage} %)',
    labelStyle: {
      color: '#000000'
    },
    border: {
      color: '#000000',
      width: 0.5
    },
  };
  public export() {
    const promise =
this.treemap!.export('PNG', 'TreeMap', 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 treemap 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 { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { TreeMap } from '@syncfusion/ej2-treemap';
import { Component, ViewChild } from '@angular/core';
import { PdfExportService } from '@syncfusion/ej2-angular-treemap';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' #treemap [allowPdfExport]=true
style='display: block;' [dataSource]='data' weightValuePath='GDP'
[leafItemSettings]='leafItemSettings'>
</ejs-treemap> <div>
<button id="togglebtn2" (click)='export()'>Export</button>
</div>`,
  providers: [PdfExportService]
})
export class AppComponent {
  @ViewChild('treemap')
  public treemap?: TreeMap;
  public data: object[] = [
    {State:"United States", GDP:17946, percentage:11.08, Rank:1},
    {State:"China", GDP:10866, percentage: 28.42, Rank:2},
    {State:"Japan", GDP:4123, percentage:-30.78, Rank:3},
    {State:"Germany", GDP:3355, percentage:-5.19, Rank:4},
    {State:"United Kingdom", GDP:2848, percentage:8.28, Rank:5},
    {State:"France", GDP:2421, percentage:-9.69, Rank:6},
    {State:"India", GDP:2073, percentage:13.65, Rank:7},
    {State:"Italy", GDP:1814, percentage:-12.45, Rank:8},
    {State:"Brazil", GDP:1774, percentage:-27.88, Rank:9},
    {State:"Canada", GDP:1550, percentage:-15.02, Rank:10}
  ];
  public leafItemSettings: object = {
    labelPath: 'State',
    labelFormat: '${State}<br>${GDP} Trillion<br>(${percentage} %)',
    labelStyle: {
      color: '#000000'
    },
    border: {
      color: '#000000',
      width: 0.5
    },
  };
  public export() {
    this.treemap!.export('PDF', 'TreeMap', 0);
  }
}

```

MAIN.TS

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

The exporting of the treemap as base64 string is not supported in the PDF export.

Accessibility in Angular TreeMap component

The TreeMap 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 TreeMap 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 TreeMap component follows the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the TreeMap component:

Attributes	Purpose
---	---
<code>role=region</code>	It specifies the TreeMap areas that do not support interactive functions like selection and highlight.
<code>role=button</code>	It specifies the TreeMap areas where interactive functions such as selection and highlight are available.
<code>aria-label</code>	Provides an accessible name for the title, subtitle, data labels, legend title, and legend item labels.

Screen reading in TreeMap

Accessibility in the TreeMap component ensures that all users, regardless of ability or disability, can use screen reading. The following TreeMap elements will be read aloud using screen reading software, such as Narrator for Windows.

Elements	Description
---	---
Data labels	Reads the labels displayed on leaf items of the TreeMap.
Title	Reads the title in the TreeMap.
Subtitle	Reads the title below the main title content in the TreeMap.
Legend title	Reads the title of the legend in the TreeMap.
Legend item label	Reads the label of the legend item in the TreeMap.

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion Angular components](#)

Internationalization in Angular Treemap component

The TreeMap control supports internationalization for the following elements:

- Data label
- Tooltip

For more information about number and date formatter, refer to [internationalization](#).

<!-- markdownlint-disable MD036 -->

Globalization

Globalization is the process of designing and developing a component that works in different cultures/locales. Internationalization library is used to globalize number, date, and time values in the tree map control using the [format](#) property in the TreeMap.

Numeric format

In the following code example, tooltip is globalized to Deutsch culture.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapTooltipService, TreeMapAllModule } from
 '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
import { setCulture, setCurrencyCode } from '@syncfusion/ej2-base';
setCulture('de');
setCurrencyCode('EUR');
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='GDP' format='c' useGroupingSeparator=
'true' [tooltipSettings]='tooltip'
>
</ejs-treemap>`,
})
export class AppComponent {
  public data: object[] = [
    { State: 'United States', GDP: 17946, percentage: 11.08, Rank: 1 },
    { State: 'China', GDP: 10866, percentage: 28.42, Rank: 2 },
    { State: 'Japan', GDP: 4123, percentage: -30.78, Rank: 3 },
    { State: 'Germany', GDP: 3355, percentage: -5.19, Rank: 4 },
    { State: 'United Kingdom', GDP: 2848, percentage: 8.28, Rank: 5 },
    { State: 'France', GDP: 2421, percentage: -9.69, Rank: 6 },
    { State: 'India', GDP: 2073, percentage: 13.65, Rank: 7 },
    { State: 'Italy', GDP: 1814, percentage: -12.45, Rank: 8 },
    { State: 'Brazil', GDP: 1774, percentage: -27.88, Rank: 9 },
    { State: 'Canada', GDP: 1550, percentage: -15.02, Rank: 10 },
  ];
  public tooltip: object = {
    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));
```

Right-to-left rendering

The TreeMap control supports right-to-left rendering for all its elements such as nodes, tooltip, data labels, and legends.

Legend with Rtl support

If set the [enableRtl](#) property to **true**, then the legend icon will be rendered on the right and the legend text will be rendered on the left of the legend icon.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;'
[dataSource]='data' weightValuePath='count' [legendSettings]='legendSettings'
colorValuePath= 'color' enableRtl='true'
[leafItemSettings]='leafItemSettings' >
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [{ Car:'Mustang', Brand:'Ford', count:232, color:
'#71B081' },
{ Car:'EcoSport', Brand:'Ford', count:121, color:
'#5A9A77' },
{ Car:'Swift', Brand:'Maruti', count:143, color:
'#498770' },
{ Car:'Baleno', Brand:'Maruti', count:454, color:
'#39776C' },
{ Car:'Vitara Brezza', Brand:'Maruti', count:545 ,
color: '#266665' },
{ Car:'A3 Cabriolet', Brand:'Audi',count:123, color:
'#124F5E' }
];
  public legendSettings: object= {
    visible: true,
    position:'Top'
  };
  public leafItemSettings:object= {
    labelPath: 'Car'
  };
}
```


MAIN.TS

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

Tooltip with Rtl support

If the [enableRtl](#) property is set to **true**, the tooltip data will be rendered in reverse direction.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;' [palette]
='palette' [dataSource]='data' weightValuePath='count'
[tooltipSettings]='tooltipSettings' colorValuePath = 'color' enableRtl
='true'
[leafItemSettings]='leafItemSettings' >
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [{ fruit:'Apple', count:5000 },
    { fruit:'Mango', count:3000 },
    { fruit:'Orange', count:2300 },
    { fruit:'Banana', count:500 },
    { fruit:'Grape', count:4300 },
    { fruit:'Papaya', count:1200 },
    { fruit:'Melon', count:4500 }
  ];
  public tooltipSettings: object= {
    visible: true,
    format:'${count} : ${fruit}'
  };
  public leafItemSettings:object= {
    labelPath: 'fruit'
  };
  public palette:object =['#71B081', '#5A9A77', '#498770', '#39776C',
'#266665', '#124F5E'];
}
```

MAIN.TS

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

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

Treemap Item Rendering Direction

The direction of TreeMap item is **TopLeftBottomRight** by default and customize the rendering direction of the TreeMap item by setting the [renderDirection](#) property.

The TreeMap can be rendered in the following directions:

- TopLeftBottomRight
- TopRightBottomLeft
- BottomRightTopLeft
- BottomLeftTopRight

The following example demonstrate, how to render the treemap in the RTL direction with **TopLeftBottomRight**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;' [palette]
='palette' [dataSource]='data' weightValuePath='count'
renderDirection='TopLeftBottomRight' >
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [{ fruit:'Apple', count:5000 },
    { fruit:'Mango', count:3000 },
    { fruit:'Orange', count:2300 },
    { fruit:'Banana', count:500 },
    { fruit:'Grape', count:4300 },
    { fruit:'Papaya', count:1200 },
    { fruit:'Melon', count:4500 }
  ];
  public leafItemSettings:object= {
    labelPath: 'fruit'
  };
  public palette:object =['#71B081', '#5A9A77', '#498770', '#39776C',
    '#266665', '#124F5E'];
}
```

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 example demonstrate, how to render the treemap in the RTL direction with **TopRightBottomLeft**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;' [palette]
='palette' [dataSource]='data' weightValuePath='count'
renderDirection='TopRightBottomLeft' >
</ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [{ fruit:'Apple', count:5000 },
    { fruit:'Mango', count:3000 },
    { fruit:'Orange', count:2300 },
    { fruit:'Banana', count:500 },
    { fruit:'Grape', count:4300 },
    { fruit:'Papaya', count:1200 },
    { fruit:'Melon', count:4500 }
  ];
  public leafItemSettings:object= {
    labelPath: 'fruit'
  };
  public palette:object =['#71B081', '#5A9A77', '#498770', '#39776C',
    '#266665', '#124F5E'];
}
```

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 example demonstrate, how to render the treemap in the RTL direction with **BottomRightTopLeft**.

APP.COMPONENT.TS

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

```

import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-treemap id='container' style='display: block;' [palette]
='palette' [dataSource]='data' weightValuePath='count'
  renderDirection='BottomRightToLeft' >
    </ejs-treemap>`
})
export class AppComponent {
  public data: object[] = [{ fruit:'Apple', count:5000 },
    { fruit:'Mango', count:3000 },
    { fruit:'Orange', count:2300 },
    { fruit:'Banana', count:500 },
    { fruit:'Grape', count:4300 },
    { fruit:'Papaya', count:1200 },
    { fruit:'Melon', count:4500 }
  ];
  public leafItemSettings:object= {
    labelPath: 'fruit'
  };
  public palette:object =['#71B081', '#5A9A77', '#498770', '#39776C',
    '#266665', '#124F5E'];
}

```

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 example demonstrate, how to render the treemap in the RTL direction with **BottomLeftToRight**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapLegendService, TreeMapTooltipService,
TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component } from '@angular/core';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  providers: [TreeMapLegendService, TreeMapTooltipService],
  standalone: true,
  selector: 'app-container',

```

```

    template: `<ejs-treemap id='container' style='display: block;' [palette]
    ='palette' [dataSource]='data' weightValuePath='count'
    renderDirection='BottomLeftTopRight' >
    </ejs-treemap>`
  })
  export class AppComponent {
    public data: object[] = [{ fruit:'Apple', count:5000 },
      { fruit:'Mango', count:3000 },
      { fruit:'Orange', count:2300 },
      { fruit:'Banana', count:500 },
      { fruit:'Grape', count:4300 },
      { fruit:'Papaya', count:1200 },
      { fruit:'Melon', count:4500 }
    ];
    public leafItemSettings:object= {
      labelPath: 'fruit'
    };
    public palette:object =['#71B081', '#5A9A77', '#498770', '#39776C',
    '#266665', '#124F5E'];
  }

```

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 Treemap component

This article describes the API migration process of Accordion component from Essential JS 1 to Essential JS 2.

Data Binding

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| DataSource | **Property:** `dataSource`
 Ts: `@Component({templateUrl: <ej-treemap id="treemap" [dataSource]="dataSource"></ej-treemap>})`
 export class DefaultComponent {constructor(public treemapDataService: TreeMapDataService)
 {

dataSource:any= treemapDataService.getTreemapData();
 }
 }
Property: `dataSource`
 Ts: `@Component({templateUrl: <ejs-treemap id="treemap" [weightValuePath]='weightValuePath' [dataSource]="dataSource"></ejs-treemap>})`
 export class DefaultComponent {

weightValuePath: string = 'Population';dataSource: object[] = [{ Continent: "Asia", population: 1749046000}];
 }

Appearance

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

```
|Layout| Property: itemsLayoutMode<br/><br/> <ej-treemap id="treemap" itemsLayoutMode=
"sliceanddiceauto" ></ej-treemap>|Property: layoutType<br/><br/> <ejs-treemap id="treemap"
layoutType='SliceAndDiceAuto'></ejs-treemap>|
```

```
|Weight Value Path| Property: weightValuePath<br/><br/> <ej-treemap id="treemap"
weightValuePath= "Population" ></ej-treemap>|Property: weightValuePath<br/><br/><ejs-
treemap id="treemap" weightValuePath= 'Population' ></ejs-treemap>|
```

```
|Range Color Value Path| Property: colorValuePath<br/><br/> <ej-treemap id="treemap"
colorValuePath= "Continent"></ej-treemap>|Property: rangeColorValuePath<br/><br/> <ejs-
treemap id="treemap" rangeColorValuePath= 'Continent' ></ejs-treemap>|
```

```
| Equal Color Value Path | Not Applicable | Property: equalColorValuePath <br/> <br/> <ejs-treemap  
id="treemap" equalColorValuePath= 'Asia' ></ejs-treemap> |
```

```
|Height| Property: height<br/><br/> <ej-treemap id="treemap" height= 50 ></ej-  
treemap>|Property: height<br/><br/> <ejs-treemap id="treemap" height='50px' ></ejs-  
treemap>|
```

```
|Width| Property: width<br><br><ej-treemap id="treemap" width= 400 ></ej-  
treemap>|Property: width<br><br><ejs-treemap id="treemap" width= '400px' ></ejs-  
treemap>|
```

|Theme| Not Applicable| **Property:** *theme*

 <ejs-treemap id="treemap" theme='Highcontrast' ></ejs-treemap>|

```
|Localization| Property: locale<br/><br/> <ej-treemap id="treemap" locale= "en-US"></ej-treemap>|Property: locale<br/><br/> <ejs-treemap id="treemap" locale= 'en-US' ></ejs-treemap>|
```

```
| Palette Colors| Property: paletteColorMapping.colors<br/><br/> <ej-treemap id="treemap">
[paletteColorMapping]= "paletteColorMapping"></ej-
treemap><br><br/>paletteColorMapping:any={ colors: ['red','green'] }| Property:
palette<br/><br/> <ejs-treemap id="treemap" [palette]="palette"></ejs-
treemap><br><br/>palette: string[] =['red','green']|
```

|Margin| Not Applicable|**Property:** *margin*

<ejs-treemap id="treemap"
[margin]="margin"></ejs-treemap>

margin:object={ left: 5, top: 8 }|

Resize	Property: <code>enableResize</code>	<code><ej-treemap id="treemap" enableResize=" true" >/ej-treemap></code>	Not Applicable
--------	--	---	----------------

Responsive	Property: <i>isResponsive</i>	<code><ej-treemap id="treemap" isResponsive="true"></ej-treemap></code>	Not Applicable
------------	--------------------------------------	---	----------------

Leaf Items

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

— — — — —

| Border Color | **Property:** `leafItemSettings.borderBrush`
 | leafItemSettings |= "leaf"></ej-treemap>

leaf: any={ showLabels: true, borderBrush:

"blue" } | **Property:** *leafItemSettings.border*

 <ej-treemap id="treemap"
[leafItemSettings]="leafItem" ></ej-treemap>

leafItem: object={border: { color:
'white' } }|

| **Border Width** | **Property:** *leafItemSettings.borderThickness*

 <ej-treemap id="treemap"
[leafItemSettings]="leaf" ></ej-treemap>

leaf: any={showLabels: true,borderThickness:
5}| **Property:** *leafItemSettings.border*

 <ej-treemap id="treemap"
[leafItemSettings]="leafItem" ></ej-treemap>

leafItem: object={border: { width: 5 } }|

| **Gap Value** | **Property:** *leafItemSettings.gap*

 <ej-treemap id="treemap"
[leafItemSettings]="leaf" ></ej-treemap>

leaf: any={showLabels: true,gap: 5 }| **Property:**
leafItemSettings.gap

 <ej-treemap id="treemap" [leafItemSettings]="leafItem" ></ej-
treemap>

leafItem: object={gap: 5}|

| **Leaf Item Label** | **Property:** *leafItemSettings.itemTemplate*

 <ej-treemap id="treemap"
[leafItemSettings]="leaf" ></ej-treemap>

leaf: any={showLabels: true, itemTemplate:
"template"}| **Property:** *leafItemSettings.labelTemplate*

 <ej-treemap id="treemap"
[leafItemSettings]="leafItem" ></ej-treemap>

leafItem: object={labelTemplate:
'template'}|

| **Leaf Label Path** | **Property:** *leafItemSettings.labelPath*

 <ej-treemap id="treemap"
[leafItemSettings]="leaf" ></ej-treemap>

leaf: any={showLabels: true, labelPath:
"GameName"}| **Property:** *leafItemSettings.labelPath*

 <ej-treemap id="treemap"
[leafItemSettings]="leafItem" ></ej-treemap>

leafItem: object={labelPath:
'GameName'}|

| **Leaf Label Position** | **Property:** *leafItemSettings.labelPosition*

 <ej-treemap id="treemap"
[leafItemSettings]="leaf" ></ej-treemap>

leaf: any={ showLabels: true, labelPosition:
"topcenter"}| **Property:** *leafItemSettings.labelPosition*

 <ej-treemap id="treemap"
[leafItemSettings]="leafItem" ></ej-treemap>

leafItem: object={labelPosition:
'Center'}|

| **Leaf Label Color** | Not Applicable | **Property:** *leafItemSettings.fill*

 <ej-treemap
id="treemap" [leafItemSettings]="leafItem" ></ej-treemap>

leafItem: object={fill:
'red'}|

| **Random Colors** | Not Applicable | **Property:** *leafItemSettings.autoFill*

 <ej-treemap
id="treemap" [leafItemSettings]="leafItem" ></ej-treemap>

leafItem:
object={autoFill: true }|

| **Format** | Not Applicable | **Property:** *leafItemSettings.labelFormat*

 <ej-treemap
id="treemap" [leafItemSettings]="leafItem" ></ej-treemap>

leafItem: object={
labelFormat: '\$ {Continent}\$ {Population}'}|

| **Labels Visibility** | **Property:** *leafItemSettings.showLabels*

 <ej-treemap id="treemap"
[leafItemSettings]="leaf" ></ej-treemap>

leaf: any={showLabels: true }| **Property:**
leafItemSettings.showLabels

 <ej-treemap id="treemap" [leafItemSettings]="leafItem"
></ej-treemap>

leafItem: object={ showLabels: false}|

| Opacity | Not Applicable | **Property:** *leafItemSettings.opacity*

 <ejs-treemap id="treemap" [leafItemSettings]="leafItem" ></ejs-treemap>

 leafItem: object={opacity: 0.7}|

| Padding | Not Applicable | **Property:** *leafItemSettings.padding*

 <ejs-treemap id="treemap" [leafItemSettings]="leafItem" ></ejs-treemap>

 leafItem: object={padding: 5}|

| Font Customization | Not Applicable | **Property:** *leafItemSettings.labelStyle*

 <ejs-treemap id="treemap" [leafItemSettings]="leafItem" ></ejs-treemap>

 leafItem: object={labelStyle: { size: '12px', color: 'red', opacity: 0.5 }}|

| Position of Template | Not Applicable | **Property:** *leafItemSettings.templatePosition*

 <ejs-treemap id="treemap" [leafItemSettings]="leafItem" ></ejs-treemap>

 leafItem: object={templatePosition: 'Center'}|

Legend

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Legend Alignment | **Property:** *legendSettings.alignment*

 <ej-treemap id="treemap" [legendSettings]="legend" [showLegend]="true"></ej-treemap>

 legend: any={ alignment: "far"}| **Property:** *legendSettings.alignment*

 <ejs-treemap id="treemap" [legendSettings]="legend" ></ejs-treemap>

 TreeMap.Inject(TreeMapLegend);

 legend: object={ legendSettings: { alignment: 'Near'}}|

| Legend Visibility | **Property:** *showLegend*

 <ej-treemap id="treemap" [showLegend]="false"></ej-treemap>| **Property:** *legendSettings.visible*

 <ejs-treemap id="treemap" [legendSettings]="legend" ></ejs-treemap>

 TreeMap.Inject(TreeMapLegend);

 legend: object={visible: true}|

| Legend Position | **Property:** *legendSettings.dockPosition*

 <ej-treemap id="treemap" [legendSettings]="legend" [showLegend]="true"></ej-treemap>

 legend: any={dockPosition: "bottom"}| **Property:** *legendSettings.position*

 <ejs-treemap id="treemap" [legendSettings]="legend" ></ejs-treemap>

 TreeMap.Inject(TreeMapLegend);

 legend: object={position: 'Top' }|

| Legend Height | **Property:** *legendSettings.height*

 <ej-treemap id="treemap" [legendSettings]="legend" [showLegend]="true"></ej-treemap>

 legend: any={height: 40}| **Property:** *legendSettings.height*

 <ejs-treemap id="treemap" [legendSettings]="legend" ></ejs-treemap>

 TreeMap.Inject(TreeMapLegend);

 legend: object={ height: '40px' }|

| Legend Width | **Property:** *legendSettings.width*

 <ej-treemap id="treemap" [legendSettings]="legend" [showLegend]="true"></ej-treemap>

 legend: any={width: 100 }| **Property:** *legendSettings.width*

 <ejs-treemap id="treemap" [legendSettings]="legend" ></ejs-treemap>

 TreeMap.Inject(TreeMapLegend);

 legend: object={ width: '100px' }|

|Shape Height| **Property:** *legendSettings.iconHeight*

 <ej-treemap id="treemap"
[legendSettings]="legend" [showLegend]="true"></ej-treemap>

legend:
any={iconHeight: 15 }|**Property:** *legendSettings.shapeHeight*

 <ejs-treemap id="treemap"
[legendSettings]="legend" ></ejs-
treemap>

TreeMap.Inject(TreeMapLegend);

legend: object={ shapeHeight:
'40px' }|

|Shape Width| **Property:** *legendSettings.iconWidth*

 <ej-treemap id="treemap"
[legendSettings]="legend" [showLegend]="true"></ej-treemap>

legend:
any={iconWidth: 8 }|**Property:** *legendSettings.shapeWidth*

 <ejs-treemap id="treemap"
[legendSettings]="legend" ></ejs-
treemap>

TreeMap.Inject(TreeMapLegend);

legend: object={shapeWidth:
'40px'}|

|Padding| Not Applicable| **Property:** *legendSettings.shapePadding*

 <ejs-treemap
id="treemap" [legendSettings]="legend" ></ejs-
treemap>

TreeMap.Inject(TreeMapLegend);

legend: object={shapePadding:
10}|

|Legend Title| **Property:** *legendSettings.title*

 <ej-treemap id="treemap"
[legendSettings]="legend" [showLegend]="true"></ej-treemap>

legend: any={title:
"Population" }|**Property:** *legendSettings.title*

 <ejs-treemap id="treemap"
[legendSettings]="legend" ></ejs-
treemap>

TreeMap.Inject(TreeMapLegend);

legend: object={title: 'Legend' }|

|Legend Shape| Not Applicable| **Property:** *legendSettings.shape*

 <ejs-treemap
id="treemap" [legendSettings]="legend" ></ejs-
treemap>

TreeMap.Inject(TreeMapLegend);

legend: object={ shape:
'Rectangle' }|

|Legend Mode| **Property:** *legendSettings.mode*

 <ej-treemap id="treemap"
[legendSettings]="legend" [showLegend]="true"></ej-treemap>

legend: any={mode:
"interactive" }|**Property:** *legendSettings.mode*

 <ejs-treemap id="treemap"
[legendSettings]="legend" ></ejs-
treemap>

TreeMap.Inject(TreeMapLegend);

legend: object={mode:
'Interactive'}|

|Legend Text Customization| Not Applicable| **Property:** *legendSettings.textStyle*

 <ejs-
treemap id="treemap" [legendSettings]="legend" ></ejs-
treemap>

TreeMap.Inject(TreeMapLegend);

legend: object={textStyle: { size:
'10px', opacity: 0.5, color: 'red' } }|

|Legend Title Customization| Not Applicable| **Property:** *legendSettings.titleStyle*

 <ejs-
treemap id="treemap" [legendSettings]="legend" ></ejs-
treemap>

TreeMap.Inject(TreeMapLegend);

legend: object={titleStyle: { size:
'10px', opacity: 0.5, color: 'red' } }|

| Legend Shape Border | Not Applicable | **Property:** *legendSettings.shapeBorder*

 <ej-treemap id="treemap" [legendSettings]="legend" >/ej-treemap>

 TreeMap.Inject(TreeMapLegend);

 legend: object={shapeBorder: {width: 2, color: 'red' }}|

| Legend Template | **Property:** *legendSettings.template*

 <ej-treemap id="treemap" [legendSettings]="legend" [showLegend]="true" >/ej-treemap>

 legend: any={template: "template"}| Not Applicable|

| Left Label | **Property:** *legendSettings.leftLabel*

 <ej-treemap id="treemap" [legendSettings]="legend" [showLegend]="true" >/ej-treemap>

 legend: any={} | Not Applicable|

| Right Label | **Property:** *legendSettings.rightLabel*

 <ej-treemap id="treemap" [legendSettings]="legend" [showLegend]="true" >/ej-treemap>

 legend: any={} | Not Applicable|

| Legend Shape Image | Not Applicable | **Property:** *legendSettings.imageUrl*

 <ej-treemap id="treemap" [legendSettings]="legend" >/ej-treemap>

 TreeMap.Inject(TreeMapLegend);

 legend: object={imageUrl: "image.png"}|

| Position in Intractive Legend | Not Applicable | **Property:** *legendSettings.labelPosition*

 <ej-treemap id="treemap" [legendSettings]="legend" >/ej-treemap>

 TreeMap.Inject(TreeMapLegend);

 legend: object={labelPosition: "Center" }|

| Legend Location | Not Applicable | **Property:** *legendSettings.location*

 <ej-treemap id="treemap" [legendSettings]="legend" >/ej-treemap>

 TreeMap.Inject(TreeMapLegend);

 legend: object={location: { x: 10, y: 20 }}|

| Legend Orientation | Not Applicable | **Property:** *legendSettings.orientation*

 <ej-treemap id="treemap" [legendSettings]="legend" >/ej-treemap>

 TreeMap.Inject(TreeMapLegend);

 legend: object={orientation: "Horizontal" }|

Levels

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Random Colors | Not Applicable | **Property:** *levels.autoFill*

 <ej-treemap id="treemap"> <e-levels><e-level autoFill="true">/e-level></e-levels></ej-treemap>|

| Level Background Color | **Property:** *levels.groupBackground*

 <ej-treemap id="treemap"> <e-levels><e-level groupBackground= "white" >/e-level></e-levels></ej-treemap>| **Property:** *levels.fill*

 <ej-treemap id="treemap"> <e-levels><e-level fill= 'white'>/e-level> </e-levels></ej-treemap>|

| Level Border Color | **Property:** *levels.groupBorderColor*
`<ej-treemap id="treemap"> <e-levels><e-level groupBorderColor="#58585B"></e-level></e-levels></ej-treemap>` | **Property:** *levels.border.color*
`<ejs-treemap id="treemap"><e-levels><e-level [border]="border"></e-level></e-levels></ejs-treemap>`
`border:object={color: "#58585B"}|`

| Level Border Width | **Property:** *levels.groupBorderThickness*
`<ej-treemap id="treemap"><e-levels><e-level groupBorderThickness= 2></e-level></e-levels></ej-treemap>` | **Property:** *levels.border.width*
`<ejs-treemap id="treemap"> <e-levels><e-level [border]="border"> </e-level> </e-levels></ejs-treemap>`
`border:object={width: 2}|`

| Group Gap | **Property:** *levels.groupGap*
`<ej-treemap id="treemap"><e-levels><e-level groupGap= 2 ></e-level></e-levels></ej-treemap>` | **Property:** *levels.groupGap*
`<ej-treemap id="treemap"><e-levels><e-level groupGap= 2 ></e-level></e-levels></ej-treemap>` |

| Group Padding | **Property:** *levels.groupPadding*
`<ej-treemap id="treemap"><e-levels><e-level groupPadding= 1 ></e-level></e-levels></ej-treemap>` | **Property:** *levels.groupPadding*
`<ejs-treemap id="treemap"><e-levels><e-level groupPadding= 1></e-level> </e-levels></ejs-treemap>` |

| Group Path | **Property:** *levels.groupPath*
`<ej-treemap id="treemap"><e-levels><e-level groupPath="pathname"></e-level></e-levels></ej-treemap>` | **Property:** *levels.groupPath*
`<ejs-treemap id="treemap"><e-levels><e-level groupPath='pathname'> </e-level></e-levels></ejs-treemap>` |

| Height of Header Level | **Property:** *levels.headerHeight*
`<ej-treemap id="treemap"><e-levels><e-level headerHeight= 20 ></e-level> </e-levels></ej-treemap>` | **Property:** *levels.headerHeight*
`<ej-treemap id="treemap"> <e-levels><e-level headerHeight= 20></e-level></e-levels></ej-treemap>` |

| Header Template | **Property:** *levels.headerTemplate*
`<ej-treemap id="treemap"> <e-levels><e-level headerTemplate= "template"></e-level></e-levels></ej-treemap>` | **Property:** *levels.headerTemplate*
`<ejs-treemap id="treemap"> <e-levels><e-level headerTemplate= 'template' ></e-level></e-levels></ejs-treemap>` |

| Opacity of Color | Not Applicable | **Property:** *levels.opacity*
`<ej-treemap id="treemap"><e-levels><e-level opacity=0.5></e-level></e-levels></ej-treemap>` |

| Header Visibility | **Property:** *levels.showHeader*
`<ej-treemap id="treemap"> <e-levels><e-level showHeader="false"></e-level> </e-levels></ej-treemap>` | **Property:** *levels.showHeader*
`<ej-treemap id="treemap"> <e-levels><e-level showHeader= "false"></e-level></e-levels></ej-treemap>` |

| Template Position | **Property:** *levels.labelPosition*
`<ej-treemap id="treemap"><e-levels><e-level labelPosition= "topleft" ></e-level></e-levels></ej-treemap>` | **Property:** *levels.templatePosition*
`<ej-treemap id="treemap"><e-levels><e-level templatePosition='Center'> </e-level></e-levels></ej-treemap>` |

| Header Style | Not Applicable | **Property:** *levels.headerStyle*

 <ejs-treemap id="treemap">
<e-levels><e-level [headerStyle]="headerStyle" ></e-level></e-levels></ejs-
treemap>

headstyle: object={ color: 'red', size: '16px', opacity: 0.7}|

| Header Format | Not Applicable | **Property:** *levels.headerFormat*

 <ejs-treemap
id="treemap"> <e-levels><e-level headerFormat= '\${{Continent}}' </e-level></e-levels></ejs-
treemap>|

| Header Alignment | Not Applicable | **Property:** *levels.headerAlignment*

 <ejs-treemap
id="treemap"><e-levels><e-level headerAlignment= 'Center' ></e-level></e-levels></ejs-
treemap>|

Selection

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Selection | **Property:** *selectionMode*

 <ej-treemap id="treemap" selectionMode=
"default"></ej-treemap>| **Property:** *selectionSettings.mode*

 <ejs-treemap id="treemap"
[selectionSettings]="selection" ></ejs-
treemap>

TreeMap.Inject(TreeMapSelection);

selection: object={enable: true,
mode: 'Item'}|

| Selection Color | Not Applicable | **Property:** *selectionSettings.fill*

 <ejs-treemap
id="treemap" [selectionSettings]="selection" ></ejs-
treemap>

TreeMap.Inject(TreeMapSelection);

selection: object={enable: true,
fill: 'blue'}|

| Selection Color Opacity | Not Applicable | **Property:** *selectionSettings.opacity*

 <ejs-treemap
id="treemap" [selectionSettings]="selection" ></ejs-
treemap>

TreeMap.Inject(TreeMapSelection);

selection: object={enable: true,
fill: 'blue', opacity: 0.6 }|

| Border for selection | Not Applicable | **Property:** *selectionSettings.border*

 <ejs-treemap
id="treemap" [selectionSettings]="selection" ></ejs-
treemap>

TreeMap.Inject(TreeMapSelection);

selection: object={border: {
color: 'red', width: 2 }}|

Highlight

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Highlight Group Selection Mode | **Property:** *highlightGroupOnSelection*

 <ej-treemap
id="treemap" highlightGroupOnSelection= "true" ></ej-treemap>| **Property:**
highlightSettings.mode

 <ejs-treemap id="treemap" [highlightSettings]="highlight"
></ejs-treemap>

TreeMap.Inject(TreeMapHighlight);

highlight:
object={enable: true, mode: 'All' }|

| Highlight Selection Mode | **Property:** *highlightOnSelection*

 <ej-treemap id="treemap"
highlightGroupOnSelection= "true" ></ej-treemap>| **Property:** *highlightSettings.mode*


```
<ejs-treemap id="treemap" [highlightSettings]="highlight" ></ejs-
treemap><br><br>TreeMap.Inject(TreeMapHighlight);<br><br>highlight: object={ enable: true,
mode: 'Item' }|
```

| Highlight Group Border Color | **Property:** *highlightGroupBorderBrush*

 <ej-treemap id="treemap" highlightGroupOnSelection= "true" highlightGroupBorderBrush= 'gray' ></ej-treemap> | **Property:** *highlightSettings.border.color*

 <ejs-treemap id="treemap" [highlightSettings]="highlight" ></ejs-treemap>

TreeMap.Inject(TreeMapHighlight);

highlight: object={ enable: true, mode: 'All', border: { color: 'gray' } }|

| Highlight Group Border Width | **Property:** *highlightGroupBorderThickness*

 <ej-treemap id="treemap" highlightGroupOnSelection= "true" highlightGroupBorderThickness= 3 ></ej-treemap> | **Property:** *highlightSettings.border.width*

 <ejs-treemap id="treemap" [highlightSettings]="highlight" ></ejs-treemap>

TreeMap.Inject(TreeMapHighlight);

highlight: object={ enable: true, mode: 'All', border: { width: 3 } }|

| Highlight Selection Border Color | **Property:** *highlightBorderBrush*

 <ej-treemap id="treemap" highlightGroupOnSelection= "true" highlightBorderBrush= 'gray' ></ej-treemap> | **Property:** *highlightSettings.border.color*

 <ejs-treemap id="treemap" [highlightSettings]="highlight" ></ejs-treemap>

TreeMap.Inject(TreeMapHighlight);

highlight: object={enable: true, mode: 'Item', border: { color: 'gray' } }|

| Highlight Selection Border Width | **Property:** *highlightBorderThickness*

 <ej-treemap id="treemap" highlightGroupOnSelection= "true" highlightBorderThickness=3></ej-treemap> | **Property:** *highlightSettings.border.width*

 <ejs-treemap id="treemap" [highlightSettings]="highlight" ></ejs-treemap>

TreeMap.Inject(TreeMapHighlight);

highlight: object={enable: true, mode: 'Item', border: { width: 3 } }|

| Highlight Color | Not Applicable | **Property:** *highlightSettings.fill*

 <ejs-treemap id="treemap" [highlightSettings]="highlight" ></ejs-treemap>

TreeMap.Inject(TreeMapHighlight);

highlight: object={enable: true, fill: 'red' }|

| Highlight Color Opacity | Not Applicable | **Property:** *highlightSettings.opacity*

 <ejs-treemap id="treemap" [highlightSettings]="highlight" ></ejs-treemap>

TreeMap.Inject(TreeMapHighlight);

highlight: object={enable: true, fill: 'red', opacity: 0.5 }|

Range ColorMapping

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| From value | **Property:** *rangeColorMapping.from*

 <ej-treemap id="treemap" colorValuePath= "Growth"><e-rangecolormapping><e-rangecolor from: 1000></e-

rangeColor></e-rangecolormapping></ej-treemap>| **Property:**

leafItemSettings.colorMapping.from

 <ejs-treemap id="treemap" rangeColorValuePath = 'Growth' [leafItemSettings]="leaf" ></ej-treemap>

leaf: object={colorMapping: [{ from: 1000 }]}|

|To value| **Property:** *rangeColorMapping.to*

<ej-treemap id="treemap" colorValuePath="Growth"><e-rangecolormapping><e-rangecolor to= 100000 ></e-rangecolor></e-rangecolormapping> </ej-treemap>| **Property:** *leafItemSettings.colorMapping.to*

<ejs-treemap id="treemap" rangeColorValuePath = 'Growth' [leafItemSettings]="leaf" ></ej-treemap>

leaf: object={colorMapping: [{ to: 100000 }]}|

|Color| **Property:** *rangeColorMapping.color*

<ej-treemap id="treemap" colorValuePath="Growth"><e-rangecolormapping><e-rangecolor color= "#77D8D8" ></e-rangecolor></e-rangecolormapping> </ej-treemap>| **Property:** *leafItemSettings.colorMapping.color*

 <ejs-treemap id="treemap" rangeColorValuePath = 'Growth' [leafItemSettings]="leaf" ></ej-treemap>

leaf: object={colorMapping: [{ color: "#77D8D8" }]}|

|Legend Label| **Property:** *rangeColorMapping.legendLabel*

<ej-treemap id="treemap" colorValuePath= "Growth"><e-rangecolormapping><e-rangecolor legendLabel= "Growth"></e-rangecolor></e-rangecolormapping></ej-treemap>| **Property:** *leafItemSettings.colorMapping.label*

 <ejs-treemap id="treemap" rangeColorValuePath = 'Growth' [leafItemSettings]="leaf" ></ej-treemap>

leaf: object={colorMapping: [{ label: "Growth" }]}|

Desaturation ColorMapping

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|From value| **Property:** *desaturationColorMapping.from*

 <ej-treemap id="treemap" colorValuePath= "Growth" [desaturationColorMapping]="desaturation"></ej-treemap>

leaf: any={ from:1000}| **Property:** *leafItemSettings.colorMapping.from*

 <ejs-treemap id="treemap" rangeColorValuePath = 'Growth' [leafItemSettings]="leaf" ></ej-treemap>

leaf: object={colorMapping: [{ from: 1000 }]}|

|To value| **Property:** *desaturationColorMapping.to*

<ej-treemap id="treemap" colorValuePath= "Growth" [desaturationColorMapping]="desaturation"></ej-treemap>

leaf: any={to:10000}| **Property:** *leafItemSettings.colorMapping.to*

 <ejs-treemap id="treemap" rangeColorValuePath = 'Growth' [leafItemSettings]="leaf" ></ej-treemap>

leaf: object={colorMapping: [{ to: 10000 }]}|

|Color| **Property:** *desaturationColorMapping.color*

 <ej-treemap id="treemap" colorValuePath= "Growth" [desaturationColorMapping]="desaturation"></ej-treemap>

leaf: any={color:"blue"}| **Property:** *leafItemSettings.colorMapping.color*

 <ejs-treemap id="treemap" rangeColorValuePath = 'Growth' [leafItemSettings]="leaf" ></ej-treemap>

leaf: object={colorMapping: [{ color:"blue" }]}|

| Value | Not Applicable | **Property:** *leafItemSettings.colorMapping.value*
 id="treemap" rangeColorValuePath = 'Growth' [leafItemSettings]="leaf" >/ej-treemap>
 leaf: object={colorMapping: [{ value="India" }]}|

| Minimum Opacity | **Property:** *desaturationColorMapping.rangeMinimum*
 id="treemap" colorValuePath= "Growth" [desaturationColorMapping]="desaturation">/ej-treemap>
 leaf: any={rangeMinimum: 1}| **Property:** *leafItemSettings.colorMapping.minOpacity*
 id="treemap" rangeColorValuePath = 'Growth' [leafItemSettings]="leaf" >/ej-treemap>
 leaf: object={colorMapping: [{ minOpacity: 0.1 }]}|

| Maximum Opacity | *desaturationColorMapping.rangeMaximum*
 id="treemap" colorValuePath= "Growth" [desaturationColorMapping]="desaturation">/ej-treemap>
 leaf: any={rangeMaximum:10}| **Property:** *leafItemSettings.colorMapping.maxOpacity*
 id="treemap" rangeColorValuePath = 'Growth' [leafItemSettings]="leaf" >/ej-treemap>
 leaf: object={colorMapping: [{ maxOpacity: 0.6 }]}|

Tooltip

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Tooltip | **Property:** *showTooltip*
 id="treemap" showTooltip= "true" >/ej-treemap> | **Property:** *tooltipSettings.visible*
 id="treemap" [tooltipSettings] = 'tooltip' >/ej-treemap>
 tooltip: object={ visible: true }|

| Tooltip Template | **Property:** *tooltipTemplate*
 id="treemap" showTooltip= "true" tooltipTemplate= 'template' >/ej-treemap> | **Property:** *tooltipSettings.template*
 id="treemap" [tooltipSettings] = 'tooltip' >/ej-treemap>
 tooltip: object={template: 'template' }|

| Tooltip Border | Not Applicable | **Property:** *tooltipSettings.border*
 id="treemap" [tooltipSettings] = 'tooltip' >/ej-treemap>
 tooltip: object={border: { color: 'red', width: 2 } }|

| Tooltip Color | Not Applicable | **Property:** *tooltipSettings.fill*
 id="treemap" [tooltipSettings] = 'tooltip' >/ej-treemap>
 tooltip: object={fill: 'gray' }|

| Tooltip Format | Not Applicable | **Property:** *tooltipSettings.format*
 id="treemap" [tooltipSettings] = 'tooltip' >/ej-treemap>
 tooltip: object={format: '\$ {Population}' }|

| Tooltip Marker Shape | Not Applicable | **Property:** *tooltipSettings.markerShapes*
 id="treemap" [tooltipSettings] = 'tooltip' >/ej-treemap>
 tooltip: object={markerShapes: 'Circle' }|

| Tooltip Color Opacity | Not Applicable | **Property:** *tooltipSettings.opacity*
 id="treemap" [tooltipSettings] = 'tooltip' >/ej-treemap>
 tooltip: object={opacity: 0.5 }|

| Tooltip Text Style | Not Applicable | **Property:** *tooltipSettings.textStyle*

 <ejs-treemap id="treemap" [tooltipSettings] = 'tooltip' ></ejs-treemap>

 tooltip: object={ textStyle: { Color: 'red', opacity: 0.5, size: '12px' }}|

Drilldown

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Drilldown | **Property:** *enableDrillDown*

 <ej-treemap id="treemap" enableDrillDown="true" ></ej-treemap>| **Property:** *enableDrillDown*

 <ejs-treemap id="treemap" enableDrillDown="true" ></ejs-treemap>|

| Drilldown Level | **Property:** *drillDownLevel*

 <ej-treemap id="treemap" drillDownLevel= 1 ></ej-treemap>| **Property:** *InitialDrillSettings.groupIndex*

 <ejs-treemap id="treemap" [InitialDrillSettings]=initialDrill ></ejs-treemap>

 initialDrill: object={ groupIndex: 1 }|

Methods

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Treemap Refresh Method | **Method:** *refresh*

 var treemap = \$("#container").ejTreeMap("instance"); treemap.refresh();| **Method:** *refresh*

 var treemap = document.getElementById('container').ej2_instances[0]; treemap.refresh();|

| Method to Drilldown | **Method:** *drillDown*

 var treemap = \$("#container").ejTreeMap("instance"); treemap.drillDown();| Not Applicable|

| Append to Method | Not Applicable | **Method:** *appendTo*

 var treemap = document.getElementById('container').ej2_instances[0]; treemap.appendTo();|

| Add Event Listener Method | Not Applicable | **Method:** *addEventListener*

 var treemap = document.getElementById('container').ej2_instances[0]; treemap.addEventListener();|

| Treemap Destroy Method | Not Applicable | **Method:** *destroy*

 var treemap = document.getElementById('container').ej2_instances[0]; treemap.destroy();|

| Treemap Exporting Method | Not Applicable | **Method:** *export*

 var treemap = document.getElementById('container').ej2_instances[0]; treemap.export();|

| Get the Module Name | Not Applicable | **Method:** *getModuleName*

 var treemap = document.getElementById('container').ej2_instances[0]; treemap.getModuleName();|

| Printing the Treemap | Not Applicable | **Method:** *print*

 var treemap = document.getElementById('container').ej2_instances[0]; treemap.print();|

| Resizing the Treemap | Not Applicable | **Method:** *resizeOnTreeMap*

 var treemap = document.getElementById('container').ej2_instances[0]; treemap.resizeOnTreeMap();|

| Inject Method (Tooltip) | Not Applicable | **Method:** *resizeOnTreeMap*

 TreeMap.Inject(TreeMapTooltip);
 TreeMap.Inject(TreeMapTooltip); var treemap = document.getElementById('container').ej2_instances[0]; treemap.resizeOnTreeMap();|

| Remove Event Listener Method | Not Applicable | **Method:** *removeEventListener*
 var
 treemap = document.getElementById('container').ej2_instances[0];
 treemap.removeEventListener(); |

Events

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Treemap Load Event | Not Applicable | **Event:** *load*
 <ej-treemap id="treemap"
 (load)='load(\$event)'></ej-treemap>

public load = (args: ILoadEventArgs) => {} |

| Treemap Loaded Event | Not Applicable | **Event:** *loaded*
 <ej-treemap id="treemap"
 (loaded)='loaded(\$event)'></ej-treemap>

public loaded = (args: ILoadedEventArgs) => {} |

| Event Before Print | Not Applicable | **Event:** *beforePrint*
 <ej-treemap id="treemap"
 (beforePrint)='beforePrint(\$event)'></ej-treemap>

public beforePrint = (args: IPrintEventArgs) => {} |

| Click Event | **Event:** *click*
 <ej-treemap id="treemap" (click)="Click(\$event)"></ej-treemap>

Click(sender) {} | **Event:** *click*
 <ej-treemap id="treemap"
 (click)='click(\$event)'></ej-treemap>

public click = (args: IItemClickEventArgs) => {} |

| Drill Start Event | **Event:** *drillStarted*
 <ej-treemap id="treemap"
 (drillStarted)="drillStarted(\$event)"></ej-treemap>

drillStarted(sender) {} | **Event:** *drillStart*
 <ej-treemap id="treemap" (drillStart)='drillStart(\$event)'></ej-treemap>

public drillStart = (args: IDrillStartEventArgs) => {} |

| Drill End Event | Not Applicable | **Event:** *drillEnd*
 <ej-treemap id="treemap" (drillEnd)='drillEnd(\$event)'></ej-treemap>

public drillEnd = (args: IDrillEndEventArgs) => {} |

| Event on Item Click | Not Applicable | **Event:** *itemClick*
 <ej-treemap id="treemap"
 (itemClick)='itemClick(\$event)'></ej-treemap>

public itemClick = (args: IItemClickEventArgs) => {} |

| Treemap Item Select Event | **Event:** *treeMapItemSelected*
 <ej-treemap id="treemap"
 (treeMapItemSelected)="treeMapItemSelected(\$event)"></ej-treemap>

treeMapItemSelected(sender) {} | **Event:** *itemSelected*
 <ej-treemap id="treemap" (itemSelected)='itemSelected(\$event)'></ej-treemap>

public
 itemSelected = (args: IItemSelectedEventArgs) => {} |

| Treemap Item Rendering Event | **Event:** *itemRendering*
 <ej-treemap id="treemap"
 (itemRendering)="itemRendering(\$event)"></ej-treemap>

itemRendering(sender) {} | **Event:** *itemRendering*
 <ej-treemap id="treemap"
 (itemRendering)='itemRendering(\$event)'></ej-treemap>

public itemRendering =
 (args: IItemRenderingEventArgs) => {} |

| Treemap Item Move Event | Not Applicable | **Event:** *itemMove*
 <ej-treemap
 id="treemap" (itemMove)='itemMove(\$event)'></ej-treemap>

public itemMove =
 (args: IItemMoveEventArgs) => {} |

|Treemap Item Highlight Event| Not Applicable| **Event:** *itemHighlight*

 <ejs-treemap id="treemap" (itemHighlight)='itemHighlight(\$event)'></ejs-treemap>

public itemHighlight = (args: IItemHighlightEventArgs) => {}|

|Template Header Render Event| **Event:** *headerTemplateRendering*

<ej-treemap id="treemap" (headerTemplateRendering)="headerTemplateRendering(\$event)"></ej-treemap>

headerTemplateRendering(sender) {}| Not Applicable|

|Drilldown Item Select Event| **Event:** *drillDownItemSelected*

 <ej-treemap id="treemap" (drillDownItemSelected)="drillDownItemSelected(\$event)"></ej-treemap>

drillDownItemSelected(sender) {}| Not Applicable|

|Refresh Event| **Event:** *refreshed*

 <ej-treemap id="treemap" (refreshed)="refreshed(\$event)"></ej-treemap>

refreshed(sender) {}| Not Applicable|

|Group Select Event| **Event:** *treeMapGroupSelected*

 <ej-treemap id="treemap" (treeMapGroupSelected)="treeMapGroupSelected(\$event)"></ej-treemap>

treeMapGroupSelected(sender) {}| Not Applicable|

|Mouse Event| Not Applicable| **Event:** *mouseMove*

 <ejs-treemap id="treemap" (mouseMove)='mouseMove(\$event)'></ejs-treemap>

public mouseMove = (args: IMouseMoveEventArgs) => {}|

|Resize Event| Not Applicable| **Event:** *resize*

 <ejs-treemap id="treemap" (resize)='resize(\$event)'></ejs-treemap>

public resize = (args: IResizeEventArgs) => {}|

|Tooltip Render Event| Not Applicable| **Event:** *tooltipRendering*

 <ejs-treemap id="treemap" (tooltipRendering)='tooltipRendering(\$event)'></ejs-treemap>

public tooltipRendering = (args: ITreeMapTooltipRenderEventArgs) => {}|

|Double Click Event| **Event:** *doubleClick*

 <ej-treemap id="treemap" (doubleClick)="doubleClick(\$event)"></ej-treemap>

doubleClick(sender) {}| Not Applicable|

|Right Click Event| **Event:** *rightClick*

 <ej-treemap id="treemap" (rightClick)="rightClick(\$event)"></ej-treemap>

rightClick(sender) {}| Not Applicable|

How To

Drilldown in Angular Treemap component

Customize the header for treemap drilldown

You can add a header element as <div> and customize it to show the population of a particular country or continent on treemap drill-down. To customize the header for treemap drill-down, follow the given steps:

Step 1:

```
<!-- markdownlint-disable MD031 -->
```

```
<!-- markdownlint-disable MD010 -->
```

Initialize the treemap and enable the drill-down option.

```
`typescript
```

```

import { Component, ViewEncapsulation } from '@angular/core';
import { TreeMap, IDrillEventArgs, ILoadEventArgs } from '@syncfusion/ej2-angular-treemap';
import { DrillDown } from './datasource';

@Component({
  selector: 'app-container',

  template: '<ejs-treemap id="container" #treemap style="display:block;" [dataSource]="dataSource"
    [weightValuePath]="weightValuePath"[leafItemSettings]="leafItemSettings" [palette]="palette"
    format="n" useGroupingSeparator="true" enableDrillDown="true"><e-levels><e-level
    groupPath="Continent" fill="#336699" [border]="border"> </e-level><e-level groupPath="States"
    fill="#336699" [border]="border"> </e-level><e-level groupPath="Region" showHeader="false"
    fill="#336699" [border]="border"></e-level></e-levels></ejs-treemap>',

  encapsulation: ViewEncapsulation.None
})

export class AppComponent implements OnInit {
  public palette: string[] = ['#9999ff', '#CCFF99', '#FFFF99', '#FF9999', '#FF99FF', '#FFCC66'];
  public dataSource: object[] = DrillDown;
  public weightValuePath: string = 'Population';
  public leafItemSettings: object = {
    labelPath: 'Name',
    showLabels: false,
    labelStyle: { size: '0px' },
    border: { color: 'black', width: 0.5 }
  };
  border: object = {
    color: 'black',
    width: 0.5
  };
};

```

Step 2:

Show the population of a particular continent in the treemap **loaded** event. In this event, you can get the header element.

```

`typescript

public loaded = (args: ILoadEventArgs) => {
  let header: Element = document.getElementById('header');

```

```

let population: number = 0;
for (let i: number = 0; i < args.treemap.layout.renderItems[0]['parent'].Continent.length; i++) {
  population += +(args.treemap.layout.renderItems[0]['parent'].Continent[i]['data'].Population);
}

header.innerHTML = 'Continent - Population : ' + population
}
,

```

Step 3:

Customize the population for drilled countries or states in the header element when drill-down the treemap. The `drillEnd` event will be triggered when treemap is drilled.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import { TreeMap, IDrillEventArgs, ILoadEventArgs } from '@syncfusion/ej2-angular-treemap';
import { DrillDown } from './datasource';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: '<ejs-treemap id="container" #treemap style="display:block;" [dataSource]="dataSource" [weightValuePath]="weightValuePath" [leafItemSettings]="leafItemSettings" [palette]="palette" format="n" useGroupingSeparator="true" enableDrillDown="true" (loaded)="loaded($event)" (drillEnd)="drillEnd($event)"><e-levels><e-level groupPath="Continent" fill="#336699" [border]="border"> </e-level><e-level groupPath="States" fill="#336699" [border]="border"> </e-level><e-level groupPath="Region" showHeader="false" fill="#336699" [border]="border"></e-level></e-levels></ejs-treemap>',
  encapsulation: ViewEncapsulation.None
})
export class AppComponent implements OnInit {
  ngOnInit(): void {
    throw new Error('Method not implemented.');
```

```

  }
  public loaded = (args: ILoadEventArgs) => {
    let header: Element = document.getElementById('header') as Element;
    let population: number = 0;
    for (let i: number = 0; i <
args.treemap!.layout.renderItems[0]['parent'].Continent.length; i++) {

```

```

        population +=
+ (args.treemap!.layout.renderItems[0]['parent'].Continent[i]['data'].Population);
    }
    header.innerHTML = 'Continent - Population : ' + population
  }
  public drillEnd = (args: IDrillEndEventArgs) => {
    let header: Element = document.getElementById('header') as Element;
    let layout: Element =
document.getElementById("container_TreeMap_Squarified_Layout") as Element;
    let population: number = 0;
    if (args.treemap!.layout.renderItems[0]['isDrilled']) {
      for (let i: number = 0; i < args.treemap!.layout.renderItems.length;
i++) {
        population +=
+ (args.treemap!.layout.renderItems[i]['data'].Population);
      }
      header.innerHTML =
layout.children[0].children[1].innerHTML.split(' ')[1] + ' - ' + population;
    }
    else if (args.treemap!.layout.renderItems[0]['parent'].Continent) {
      for (let i: number = 0; i <
args.treemap!.layout.renderItems[0]['parent'].Continent.length; i++) {
        population +=
+ (args.treemap!.layout.renderItems[0]['parent'].Continent[i]['data'].Population);
      }
      header.innerHTML = 'Continent - Population : ' + population;
    }
    else {
      population = args.treemap!.layout.renderItems[0]['data'].Population;
      header.innerHTML =
layout.children[0].children[1].innerHTML.split(' ')[1] + ' - Population : ' +
population;
    }
  }
  public palette: string[] = ['#9999ff', '#CCFF99', '#FFFF99', '#FF9999',
'#FF99FF', '#FFCC66'];
  public dataSource: object[] = DrillDown;
  public weightValuePath: string = 'Population';
  public leafItemSettings: object = {
    labelPath: 'Name',
    showLabels: false,
    labelStyle: { size: '0px' },
    border: { color: 'black', width: 0.5 }
  };
  border: object = {
    color: 'black',
    width: 0.5
  };
};
};

```

MAIN.TS

```

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

```

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

Drilldown with label in Angular Treemap component

You can add a label template as <div> element to the treemap control when using the label template. To add a label template to the treemap control, you have to hide another labels by setting the `showLabels` property to **false** in `leafItemSettings` to show only the label template.

To add label template to treemap drilldown, follow the given steps:

Step 1:

```
<!-- markdownlint-disable MD010 -->
```

Create a tree map control and enable the drill-down option.

```
`typescript
```

```
import { Component, ViewEncapsulation } from '@angular/core';
import { TreeMap, IDrillStartEventArgs } from '@syncfusion/ej2-angular-treemap';
import { CarSales } from './datasource';
/
```

- Default sample

```
*/
```

```
@Component({
  selector: 'app-container',
  template: '<ejs-treemap id="container" #treemap style="display: block;" [dataSource]="dataSource"
    [weightValuePath]="weightValuePath" enableDrillDown="true" [palette]="palette"><e-levels><e-level
    groupPath="Continent" [border]="border"></e-level><e-level groupPath="Company"
    [border]="border"> </e-level></e-levels></ejs-treemap>',
  encapsulation: ViewEncapsulation.None
})
export class TreemapDrillDownComponent {
  public weightValuePath: string = "Sales";
  public palette: string[] = ['white'];
  public dataSource: object[] = CarSales;
  public border: Object = { width: 0.5, color: 'black' }
};
`
```

Step 2:

Add the label template in the `leafItemSettings` options, and then set the `showLabels` property to **false** to hide another labels and show only label template.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TreeMapModule, TreeMapAllModule } from '@syncfusion/ej2-angular-treemap'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import { TreeMap, IDrillStartEventArgs } from '@syncfusion/ej2-angular-treemap';
import { CarSales } from '../datasource';
@Component({
  imports: [
    TreeMapModule, TreeMapAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: '<ejs-treemap id="container" #treemap style="display:block;" [dataSource]="dataSource" [weightValuePath]="weightValuePath" [leafItemSettings]="leafItemSettings" enableDrillDown="true" (drillStart)="drillStart($event)" [palette]="palette"><e-levels><e-level groupPath="Continent" [border]="border"> </e-level> <e-level groupPath="Company" [border]="border"></e-level></e-levels></ejs-treemap>',
  encapsulation: ViewEncapsulation.None
})
export class AppComponent implements OnInit {
  ngOnInit(): void {
    throw new Error('Method not implemented.');
```

```

  }
  public drillStart = (args: IDrillStartEventArgs) => {
    let labelElementGroup: HTMLElement =
```

```

    document.getElementById('container_Label_Template_Group') as HTMLElement;
    labelElementGroup.remove();
  }
```

```

  public weightValuePath: string = "Sales";
```

```

  public palette: string[] = ['white'];
```

```

  public dataSource: object[] = CarSales;
```

```

  public leafItemSettings: object = {
```

```

    showLabels: false,
```

```

    labelTemplate: '#template',
```

```

    templatePosition: 'Center'
  };
  public border: Object = { width: 0.5, color: 'black' }
```

```

};
```

MAIN.TS

```

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

TreeView

Getting started with Angular Treeview component

This section explains the steps required to create a simple [Angular TreeView](#) component, and configure its available functionalities

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 Treeview 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.

Registering Treeview Module

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

```
`typescript
```

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

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

```
// import Treeview component Module
```

```
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations';
```

```
import { AppComponent } from './app.component';
```

```
@NgModule({
```

```
//declaration of Treeview module into NgModule
```

```
imports: [ BrowserModule, TreeViewModule ],
```

```
declarations: [ AppComponent ],
```

```
bootstrap: [ AppComponent ]
```

```
})
```

```
export class AppModule { }
```

```
,
```

Adding CSS Reference

- Add Treeview component's styles as given below in `styles.css`.

```
`css
```

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

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

```
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
```

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

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

Add Treeview component

Modify the template in [src/app/app.component.ts] file to render the Treeview component.

Add the Angular Treeview by using `<ejs-treeview>` selector in `template` section of the app.component.ts file.

```
`typescript
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: <ejs-treeview id='treeelement' ></ejs-treeview>
})
export class AppComponent {}
```

Binding data source

TreeView can load data either from local data sources or remote data services. This can be done using the [dataSource](#) property that is a member of the `fields` property. The `dataSource` property supports array of JavaScript objects and `DataManager`.

Here, an array of JSON values is passed to the TreeView component.

```
`typescript
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  selector: 'app-root',
  template: <ejs-treeview id='treeelement' [fields]='field'></ejs-treeview>,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  constructor() {
  }
  // defined the array of data
  public hierarchicalData: Object[] = [
    { id: '01', name: 'Music', expanded: true,
    subChild: [
      {id: '01-01', name: 'Gouttes.mp3'},
```

```

]
},
{
  id: '02', name: 'Videos',
  subChild: [
    {id: '02-01', name: 'Naturals.mp4'},
    {id: '02-02', name: 'Wild.mpeg'}
  ]
},
{
  id: '03', name: 'Documents',
  subChild: [
    {id: '03-01', name: 'Environment Pollution.docx'},
    {id: '03-02', name: 'Global Water, Sanitation, & Hygiene.docx'},
    {id: '03-03', name: 'Global Warming.ppt'},
    {id: '03-02', name: 'Social Network.pdf'},
    {id: '03-03', name: 'Youth Empowerment.pdf'},
  ]
}
];
public field:Object={ dataSource: this.hierarchicalData, id: 'id', text: 'name', child: 'subChild' };
}
`

```

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 { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({

```

```

imports: [
    FormsModule, TreeViewModule
],
standalone: true,
selector: 'app-container',
template: `<div id='treeparent'><ejs-treeview id='treeelement'
[fields]='field'></ejs-treeview></div>`
}))
export class AppComponent {
    constructor() {
    }
    // defined the array of data
    public hierarchicalData: Object[] = [
        { id: '01', name: 'Music', expanded: true,
          subChild: [
            {id: '01-01', name: 'Gouttes.mp3'},
          ]
        },
        {
          id: '02', name: 'Videos',
          subChild: [
            {id: '02-01', name: 'Naturals.mp4'},
            {id: '02-02', name: 'Wild.mpeg'}
          ]
        },
        {
          id: '03', name: 'Documents',
          subChild: [
            {id: '03-01', name: 'Environment Pollution.docx'},
            {id: '03-02', name: 'Global Water, Sanitation, &
Hygiene.docx'},
            {id: '03-03', name: 'Global Warming.ppt'},
            {id: '03-02', name: 'Social Network.pdf'},
            {id: '03-03', name: 'Youth Empowerment.pdf'},
          ]
        }
    ];
    public field:Object = { dataSource: this.hierarchicalData, id: 'id', text:
'name', child: 'subChild' };
}

```

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 TreeView](#) example to know how to present and manipulate data.

See Also

- [How to customize treeview as accordion](#)
- [How to set tooltip for treeview nodes](#)

- [How to filter nodes in treeview](#)
- [How to get all the child nodes through parentID](#)

Data binding in Angular Treeview component

The TreeView component provides the option to load data either from local data sources or from remote data services.

This can be done through `dataSource` property that is a member of the `fields` property.

The `dataSource` property supports array of JavaScript objects and `DataManager`.

It also supports different kinds of data services such as OData, OData V4, Web API, URL, and JSON with the help of `DataManager` adaptors.

TreeView has `load on demand` (Lazy load), by default. It reduces the bandwidth size when consuming huge data.

It loads first level nodes initially, and when parent node is expanded, loads the child nodes based on the `parentID/child` member.

By default, the `loadOnDemand` is set to true. By disabling this property, all the tree nodes are rendered at the beginning itself.

You can use the `dataBound` event to perform actions.

This event will be triggered once the data source is populated in the TreeView.

Local data

To bind local data to the TreeView, you can assign a JavaScript object array to the `dataSource` property.

The TreeView component requires three fields (id, text, and parentID) to render local data source.

When mapper fields are not specified, it takes the default values as the mapping fields. Local data source can also be

provided as an instance of the `DataManager`. It supports two kinds of local data binding methods.

- Hierarchical data
- Self-referential data

Hierarchical data

TreeView can be populated with hierarchical data source that contains nested array of JSON objects.

You can directly assign hierarchical data to the `dataSource` property, and map all the field members with corresponding keys from the hierarchical data to `fields` property.

In the following example, **code**, **name**, and **countries** columns from hierarchical data have been mapped to **id**, **child**, and **text** fields, respectively.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
```

```

@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div id='treeparent'><ejs-treeview id='treeelement'
[fields]='field'></ejs-treeview></div>`
})
export class AppComponent {
  constructor() {
  }
  //define the data source
  public continents:Object[] = [
    {
      code: 'AF', name: 'Africa', countries: [
        { code: 'NGA', name: 'Nigeria' },
        { code: 'EGY', name: 'Egypt' },
        { code: 'ZAF', name: 'South Africa' }
      ]
    },
    {
      code: 'AS', name: 'Asia', expanded: true, countries: [
        { code: 'CHN', name: 'China' },
        { code: 'IND', name: 'India', selected: true },
        { code: 'JPN', name: 'Japan' }
      ]
    },
    {
      code: 'EU', name: 'Europe', countries: [
        { code: 'DNK', name: 'Denmark' },
        { code: 'FIN', name: 'Finland' },
        { code: 'AUT', name: 'Austria' }
      ]
    },
    {
      code: 'NA', name: 'North America', countries: [
        { code: 'USA', name: 'United States of America' },
        { code: 'CUB', name: 'Cuba' },
        { code: 'MEX', name: 'Mexico' }
      ]
    },
    {
      code: 'SA', name: 'South America', countries: [
        { code: 'BRA', name: 'Brazil' },
        { code: 'COL', name: 'Colombia' },
        { code: 'ARG', name: 'Argentina' }
      ]
    },
    {
      code: 'OC', name: 'Oceania', countries: [
        { code: 'AUS', name: 'Australia' },
        { code: 'NZL', name: 'New Zealand' },
        { code: 'WSM', name: 'Samoa' }
      ]
    }
  ],
  {

```

```

        code: 'AN', name: 'Antarctica', countries: [
            { code: 'BVT', name: 'Bouvet Island' },
            { code: 'ATF', name: 'French Southern Lands' }
        ]
    }
];
public field:Object = { dataSource: this.continents, id: 'code', text:
'name', child: 'countries' };
}

```

MAIN.TS

```

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

```

Self-referential Data

TreeView can be populated from self-referential data structure that contains array of JSON objects with **parentID** mapping.

You can directly assign self-referential data to the **dataSource** property, and map all the field members with corresponding keys from self-referential data to **fields** property.

In the following example, **id**, **pid**, **hasChild**, and **name** columns from self-referential data have been mapped to **id**, **parentID**, **hasChildren**, and **text** fields, respectively.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule,TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  // specifies the template string for the TreeView component with fields
  // property
  template: `<div id='treeparent'><ejs-treeview id='treeelement'
[fields]='field'></ejs-treeview></div>`
})
export class AppComponent {
  constructor() {
  }
  // defined the array of data
  public localData: Object[] = [
    { id: 1, name: 'Discover Music', hasChild: true, expanded: true },
    { id: 2, pid: 1, name: 'Hot Singles' },
    { id: 3, pid: 1, name: 'Rising Artists' },
    { id: 4, pid: 1, name: 'Live Music' },
    { id: 7, name: 'Sales and Events', hasChild: true },
  ]
}

```

```

{ id: 8, pid: 7, name: '100 Albums - $5 Each' },
{ id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
{ id: 10, pid: 7, name: 'CD Deals' },
{ id: 11, name: 'Categories', hasChild: true },
{ id: 12, pid: 11, name: 'Songs' },
{ id: 13, pid: 11, name: 'Bestselling Albums' },
{ id: 14, pid: 11, name: 'New Releases' },
{ id: 15, pid: 11, name: 'Bestselling Songs' },
{ id: 16, name: 'MP3 Albums', hasChild: true },
{ id: 17, pid: 16, name: 'Rock' },
{ id: 18, pid: 16, name: 'Gospel' },
{ id: 19, pid: 16, name: 'Latin Music' },
{ id: 20, pid: 16, name: 'Jazz' },
{ id: 21, name: 'More in Music', hasChild: true },
{ id: 22, pid: 21, name: 'Music Trade-In' },
{ id: 23, pid: 21, name: 'Redeem a Gift Card' },
{ id: 24, pid: 21, name: 'Band T-Shirts' }
];
// maps the appropriate column to fields property
public field: Object = { dataSource: this.localData, id: 'id', parentID:
'pid', text: 'name', hasChildren: 'hasChild' };
}

```

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

TreeView can also be populated from a remote data service with the help of **DataManager** component and **Query** property.

It supports different kinds of data services such as OData, OData V4, Web API, URL, and JSON with the help of **DataManager** adaptors.

You can assign service data as an instance of **DataManager** to the **dataSource** property. To interact with remote data source, you have to provide the endpoint **url**.

The **DataManager** that acts as an interface between the service endpoint and the TreeView requires the following information to interact with service endpoint properly.

- **DataManager->url**: Defines the service endpoint to fetch data.
- **DataManager->adaptor**: Defines the adaptor option. By default, **ODataAdaptor** is used for remote binding.

Adaptor is responsible for processing response and request from/to the service endpoint. The **@syncfusion/ej2-data** package provides some predefined adaptors designed to interact with service endpoints. They are,

- **UrlAdaptor**: Used to interact with remote services. This is the base adaptor for all remote based adaptors.
- **ODataAdaptor**: Used to interact with OData endpoints.
- **ODataV4Adaptor**: Used to interact with OData V4 endpoints.
- **WebApiAdaptor**: Used to interact with Web API created under OData standards.
- **WebMethodAdaptor**: Used to interact with web methods.

In the following example, **ODataV4Adaptor** is used to fetch data from remote services. The **EmployeeID**, **FirstName**, and **Title** columns from Employees table have been mapped to **id**, **text**, and **hasChildren** fields respectively for first level nodes.

The **OrderID**, **EmployeeID**, and **ShipName** columns from orders table have been mapped to **id**, **parentID**, and **text** fields respectively for second level nodes.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data'
@Component({
  imports: [
    FormsModule,TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  // specifies the template string for the TreeView component with remote
  dataSource
  template: `<div id='treeparent'><ejs-treeview id='treeelement'
[fields]='field'></ejs-treeview></div>`
})
export class AppComponent {
  constructor() {
  }
  //bind the DataManager instance to dataSource property
  public data: Object = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc',
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
  });
  //bind the Query instance to query property
  public query:Object = new
Query().from('Employees').select('EmployeeID,FirstName,Title').take(5);
  public query1:Object = new
Query().from('Orders').select('OrderID,EmployeeID,ShipName').take(5);
  //Map the fields
  public field:Object ={ dataSource: this.data, query: this.query, id:
'EmployeeID', text: 'FirstName', hasChildren: 'EmployeeID', tooltip: 'Title',
  child: { dataSource: this.data, query: this.query1, id:
'OrderID', parentID: 'EmployeeID', text: 'ShipName' }
  };
}
```

MAIN.TS

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

Check box in Angular Treeview component

The TreeView component allows you to check more than one node in TreeView without affecting the UI's appearance by enabling the [showCheckBox](#) property. When this property is enabled, checkbox appears before each TreeView node text.

- If one of the child nodes is not in a checked state, then the parent node will be in an intermediate state.
- If all the child nodes are in checked state, then the parent node's state will also be checked.
- If a parent node is checked, then all the child nodes' state will also be checked.

By default, the checkbox state of parent and child nodes are dependent on each other. If you need independent checked state, you can achieve it using the [autoCheck](#) property.

Using the [checkedNodes](#) property, you can set the nodes that need to be checked or get the ID of nodes that are currently checked in the TreeView component.

If you need to prevent the node check action for a particular node, the [nodeChecking](#) event can be used which is triggered before the TreeView node is checked/unchecked. The [nodeChecked](#) event will be triggered when the TreeView node is checked/unchecked successfully.

In the following example, the `showCheckBox` property is enabled.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild } from '@angular/core';
import { TreeViewComponent } from '@syncfusion/ej2-angular-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  // specifies the template string for the TreeView component with CheckBox
  template: `<div id='treeparent'><ejs-treeview id='treeelement'
[fields]='field' [showCheckBox]='showCheckBox'></ejs-treeview></div>`
})
export class AppComponent {
  @ViewChild('samples')
  public tree?: TreeViewComponent;
  constructor() {
  }
  // defined the array of data
```

```

public countries: Object[] = [
  { id: 1, name: 'Australia', hasChild: true, expanded: true },
  { id: 2, pid: 1, name: 'New South Wales' },
  { id: 3, pid: 1, name: 'Victoria' },
  { id: 4, pid: 1, name: 'South Australia' },
  { id: 6, pid: 1, name: 'Western Australia' },
  { id: 7, name: 'Brazil', hasChild: true },
  { id: 8, pid: 7, name: 'Paraná' },
  { id: 9, pid: 7, name: 'Ceará' },
  { id: 10, pid: 7, name: 'Acre' },
  { id: 11, name: 'China', hasChild: true },
  { id: 12, pid: 11, name: 'Guangzhou' },
  { id: 13, pid: 11, name: 'Shanghai' },
  { id: 14, pid: 11, name: 'Beijing' },
  { id: 15, pid: 11, name: 'Shantou' },
  { id: 16, name: 'France', hasChild: true },
  { id: 17, pid: 16, name: 'Pays de la Loire' },
  { id: 18, pid: 16, name: 'Aquitaine' },
  { id: 19, pid: 16, name: 'Brittany' },
  { id: 20, pid: 16, name: 'Lorraine' },
  { id: 21, name: 'India', hasChild: true },
  { id: 22, pid: 21, name: 'Assam' },
  { id: 23, pid: 21, name: 'Bihar' },
  { id: 24, pid: 21, name: 'Tamil Nadu' },
  { id: 25, pid: 21, name: 'Punjab' }
];
// maps the appropriate column to fields property
public field: Object = { dataSource: this.countries, id: 'id', parentID:
'pid', text: 'name', hasChildren: 'hasChild' };
// set the CheckBox to the TreeView
public showCheckBox: 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));

```

Checked nodes

You can get or set the checked nodes in TreeView at initial rendering and dynamically by using the [checkedNodes](#) property.

It returns the checked nodes' ID as an array.

In the following example, the **New South Wales** and **Western Australia** nodes are checked at initial rendering.

If any more nodes are checked, the checked nodes' IDs will be displayed in alert.

APP.COMPONENT.TS

```

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

```

```

import { TreeViewModel } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild } from '@angular/core';
import { TreeViewComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    FormsModule, TreeViewModel
  ],
  standalone: true,
  selector: 'app-container',
  // specifies the template string for the TreeView component with CheckBox
  template: `<div id='treeparent'><ejs-treeview #treeview=""
id='treeelement' [fields]='field' [showCheckBox]='showCheckBox'
(nodeChecked)='nodeChecked($event)'></ejs-treeview></div>`
})
export class AppComponent {
  constructor() {
  }
  @ViewChild('treeview')
  public tree?: TreeViewComponent;
  // defined the array of data
  public countries: Object[] = [
    { id: 1, name: 'Australia', hasChild: true, expanded: true },
    { id: 2, pid: 1, name: 'New South Wales', isChecked: true },
    { id: 3, pid: 1, name: 'Victoria' },
    { id: 4, pid: 1, name: 'South Australia' },
    { id: 6, pid: 1, name: 'Western Australia', isChecked: true },
    { id: 7, name: 'Brazil', hasChild: true },
    { id: 8, pid: 7, name: 'Paraná' },
    { id: 9, pid: 7, name: 'Ceará' },
    { id: 10, pid: 7, name: 'Acre' },
    { id: 11, name: 'China', hasChild: true },
    { id: 12, pid: 11, name: 'Guangzhou' },
    { id: 13, pid: 11, name: 'Shanghai' },
    { id: 14, pid: 11, name: 'Beijing' },
    { id: 15, pid: 11, name: 'Shantou' },
    { id: 16, name: 'France', hasChild: true },
    { id: 17, pid: 16, name: 'Pays de la Loire' },
    { id: 18, pid: 16, name: 'Aquitaine' },
    { id: 19, pid: 16, name: 'Brittany' },
    { id: 20, pid: 16, name: 'Lorraine' },
    { id: 21, name: 'India', hasChild: true },
    { id: 22, pid: 21, name: 'Assam' },
    { id: 23, pid: 21, name: 'Bihar' },
    { id: 24, pid: 21, name: 'Tamil Nadu' },
    { id: 25, pid: 21, name: 'Punjab' }
  ];
  // maps the appropriate column to fields property
  public field: Object = { dataSource: this.countries, id: 'id', parentID:
'pid', text: 'name', hasChildren: 'hasChild' };
  // set the CheckBox to the TreeView
  public showCheckBox: boolean = true;
  //set the checknodes to the TreeView
  public checkedNodes: string[] = ['2', '6'];
  public nodeChecked(args: any): void{
    alert("The checked node's id is: "+this.tree?.checkedNodes);
  }
}

```

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 check/uncheck the checkbox on clicking the tree node text](#)
- [How to disable the checkboxes alone in the tree nodes](#)
- [How to remove the checkbox of the parent node in treeview](#)

Node editing in Angular Treeview component

The TreeView allows you to edit nodes by setting the [allowEditing](#)

property to **true**.

To directly edit the nodes in place, **double click** the TreeView node or **select** the node and press **F2** key.

When editing is completed by focus out or by pressing the **Enter** key, the modified node's text saves automatically.

If you do not want to save the modified node's text in TreeView node, press **Escape** key. It does not save the edited text to the TreeView node.

- Node editing can also be performed programmatically by using the [beginEdit](#) method. On passing the node ID or element through this method, the edit textbox will be created for the particular node thus allowing us to edit it.
- If you need to validate or prevent editing, the [nodeEditing](#) event can be used which is triggered before the TreeView node is renamed. On successfully renaming a node the [nodeEdited](#) event will be triggered.

In the following example, the first level node's text cannot be changed, but all other level nodes' text can be changed.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild } from '@angular/core';
import { NodeCheckEventArgs, TreeViewComponent } from '@syncfusion/ej2-
angular-navigations';
@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  // specifies the template string for the TreeView component
```

```

    template: `<div id='treeparent'><ejs-treeview id='treeelement'
[fields]='field' [allowEditing]='allowEditing'
(nodeEditing)='editing($event)'></ejs-treeview></div>`
  })
  export class AppComponent {
    @ViewChild('samples')
    public tree?: TreeViewComponent;
    constructor() {
    }
    // defined the array of data
    public hierarchicalData: Object[] = [
      { id: '01', name: 'Local Disk(C:)', expanded: true,
        subChild: [
          {
            id: '01-01', name: 'Program Files',
            subChild: [
              { id: '01-01-01', name: '7-Zip' },
              { id: '01-01-02', name: 'Git' },
              { id: '01-01-03', name: 'IIS Express' },
            ]
          },
          {
            id: '01-02', name: 'Users', expanded: true,
            subChild: [
              { id: '01-02-01', name: 'Smith' },
              { id: '01-02-02', name: 'Public' },
              { id: '01-02-03', name: 'Admin' },
            ]
          },
          {
            id: '01-03', name: 'Windows',
            subChild: [
              { id: '01-03-01', name: 'Boot' },
              { id: '01-03-02', name: 'FileManager' },
              { id: '01-03-03', name: 'System32' },
            ]
          },
        ]
      },
      {
        id: '02', name: 'Local Disk(D:)',
        subChild: [
          {
            id: '02-01', name: 'Personals',
            subChild: [
              { id: '02-01-01', name: 'My photo.png' },
              { id: '02-01-02', name: 'Rental document.docx' },
              { id: '02-01-03', name: 'Pay slip.pdf' },
            ]
          },
          {
            id: '02-02', name: 'Projects',
            subChild: [
              { id: '02-02-01', name: 'ASP Application' },
              { id: '02-02-02', name: 'TypeScript Application' },
              { id: '02-02-03', name: 'React Application' },
            ]
          },
        ]
      },
    ],
  },

```

```

    ],
    {
      id: '02-03', name: 'Office',
      subChild: [
        { id: '02-03-01', name: 'Work details.docx' },
        { id: '02-03-02', name: 'Weekly report.docx' },
        { id: '02-03-03', name: 'Wish list.csv' },
      ]
    },
  ],
},
{
  id: '03', name: 'Local Disk(E:)', icon: 'folder',
  subChild: [
    {
      id: '03-01', name: 'Pictures',
      subChild: [
        { id: '03-01-01', name: 'Wind.jpg' },
        { id: '03-01-02', name: 'Stone.jpg' },
        { id: '03-01-03', name: 'Home.jpg' },
      ]
    },
    {
      id: '03-02', name: 'Documents',
      subChild: [
        { id: '03-02-01', name: 'Environment
Pollution.docx' },
        { id: '03-02-02', name: 'Global Warming.ppt' },
        { id: '03-02-03', name: 'Social Network.pdf' },
      ]
    },
    {
      id: '03-03', name: 'Study Materials',
      subChild: [
        { id: '03-03-01', name: 'UI-Guide.pdf' },
        { id: '03-03-02', name: 'Tutorials.zip' },
        { id: '03-03-03', name: 'TypeScript.7z' },
      ]
    },
  ],
}
];
// maps the appropriate column to fields property
public field: Object = { dataSource: this.hierarchicalData, id: 'id',
text: 'name', child: 'subChild' };
// enable the editing options to the TreeView
public allowEditing: boolean = true;
//Bind the nodeChecked event
public editing(args: NodeCheckEventArgs) {
  //check whether node is root node or not
  if (args.node.parentNode?.parentNode?.nodeName !== "LI") {
    args.cancel = 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

- [How to validate the text when renaming the tree node](#)
- [How to process the tree node operations using context menu](#)

Multiple selection in Angular Treeview component

Selection provides an interactive support and highlights the node that you select. Selection can be done through simple mouse down or keyboard interaction.

The TreeView also supports selection of multiple nodes by setting [allowMultiSelection](#) to **true**.

To multi-select, press and hold **CTRL** key and click the desired nodes. To select range of nodes, press and hold the **SHIFT** key and click the nodes.

In the following example, the `allowMultiSelection` property is enabled.

Multi selection is not applicable through touch interactions.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild } from '@angular/core';
import { TreeViewComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    FormsModule,TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  // specifies the template string for the TreeView component
  template: `<div id='treeparent'><ejs-treeview id='treeelement'
[fields]='field' [allowMultiSelection]='allowMultiSelection'></ejs-
treeview></div>`
})
export class AppComponent {
  @ViewChild('samples')
  public tree?: TreeViewComponent;
  constructor() {
  }
  // defined the array of data
  public countries: Object[] = [
    { id: 1, name: 'Australia', hasChild: true, expanded: true },
    { id: 2, pid: 1, name: 'New South Wales', isSelected: true },
    { id: 3, pid: 1, name: 'Victoria' },
```



```

    { id: 4, pid: 1, name: 'South Australia' },
    { id: 6, pid: 1, name: 'Western Australia', isSelected: true },
    { id: 7, name: 'Brazil', hasChild: true },
    { id: 8, pid: 7, name: 'Paraná' },
    { id: 9, pid: 7, name: 'Ceará' },
    { id: 10, pid: 7, name: 'Acre' },
    { id: 11, name: 'China', hasChild: true },
    { id: 12, pid: 11, name: 'Guangzhou' },
    { id: 13, pid: 11, name: 'Shanghai' },
    { id: 14, pid: 11, name: 'Beijing' },
    { id: 15, pid: 11, name: 'Shantou' },
    { id: 16, name: 'France', hasChild: true },
    { id: 17, pid: 16, name: 'Pays de la Loire' },
    { id: 18, pid: 16, name: 'Aquitaine' },
    { id: 19, pid: 16, name: 'Brittany' },
    { id: 20, pid: 16, name: 'Lorraine' },
    { id: 21, name: 'India', hasChild: true },
    { id: 22, pid: 21, name: 'Assam' },
    { id: 23, pid: 21, name: 'Bihar' },
    { id: 24, pid: 21, name: 'Tamil Nadu' },
    { id: 25, pid: 21, name: 'Punjab' }
  ];
  // maps the appropriate column to fields property
  public field: Object = { dataSource: this.countries, id: 'id', parentID:
'pid', text: 'name', hasChildren: 'hasChild', selected: 'isSelected' };
  // set the Multi Selection option to TreeView
  public allowMultiSelection: 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));

```

Selected nodes

You can get or set the selected nodes in TreeView at initial rendering and dynamically by using the [selectedNodes](#) property. It will return the selected node's ID as an array.

- The [nodeselecting](#) event is triggered before a node is selected/unselected which can be used to prevent the selection.
- The [nodeSelected](#) event is triggered once a node is successfully selected/unselected.

In the following example, **New South Wales** and **Western Australia** nodes are selected at initial rendering.

When a node is selected, the selected node's ID is displayed in alert.

APP.COMPONENT.TS

```

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

```

```

import { TreeViewModel } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild } from '@angular/core';
import { TreeViewComponent, NodeSelectEventArgs } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    FormsModule, TreeViewModel
  ],
  standalone: true,
  selector: 'app-container',
  // specifies the template string for the TreeView component
  template: `<div id='treeparent'><ejs-treeview #tree id='treeelement'
[fields]='field' [allowMultiSelection]='allowMultiSelection'
[selectedNodes]='selectedNodes' (nodeSelected)= 'nodeSelected($event)'></ejs-
treeview></div>`
})
export class AppComponent {
  @ViewChild('tree')
  public tree?: TreeViewComponent;
  constructor() {
  }
  // defined the array of data
  public countries: Object[] = [
    { id: 1, name: 'Australia', hasChild: true, expanded: true },
    { id: 2, pid: 1, name: 'New South Wales', isSelected: true },
    { id: 3, pid: 1, name: 'Victoria' },
    { id: 4, pid: 1, name: 'South Australia' },
    { id: 6, pid: 1, name: 'Western Australia', isSelected: true },
    { id: 7, name: 'Brazil', hasChild: true },
    { id: 8, pid: 7, name: 'Paraná' },
    { id: 9, pid: 7, name: 'Ceará' },
    { id: 10, pid: 7, name: 'Acre' },
    { id: 11, name: 'China', hasChild: true },
    { id: 12, pid: 11, name: 'Guangzhou' },
    { id: 13, pid: 11, name: 'Shanghai' },
    { id: 14, pid: 11, name: 'Beijing' },
    { id: 15, pid: 11, name: 'Shantou' },
    { id: 16, name: 'France', hasChild: true },
    { id: 17, pid: 16, name: 'Pays de la Loire' },
    { id: 18, pid: 16, name: 'Aquitaine' },
    { id: 19, pid: 16, name: 'Brittany' },
    { id: 20, pid: 16, name: 'Lorraine' },
    { id: 21, name: 'India', hasChild: true },
    { id: 22, pid: 21, name: 'Assam' },
    { id: 23, pid: 21, name: 'Bihar' },
    { id: 24, pid: 21, name: 'Tamil Nadu' },
    { id: 25, pid: 21, name: 'Punjab' }
  ];
  // maps the appropriate column to fields property
  public field: Object = { dataSource: this.countries, id: 'id', parentID:
'pid', text: 'name', hasChildren: 'hasChild' };
  // set the Multi Selection option to the TreeView
  public allowMultiSelection: boolean = true;
  //set the Selected nodes to the TreeView
  public selectedNodes: string[] = ['2', '6'];
  //Bind the nodeSelected event
  public nodeSelected(e: NodeSelectEventArgs) {

```

```

        alert("The selected node's id: " + this.tree?.selectedNodes); // To
        alert the selected node's id.
    };
}

```

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 hover and select the multiple line tree nodes](#)
- [How to select only one child at a time, out of one specific parent](#)

Drag and drop in Angular Treeview component

The TreeView component allows you to drag and drop any node by setting [allowDragAndDrop](#) to **true**. Nodes can be dragged and dropped at all levels of the same TreeView.

The dragged nodes can be dropped at any level by indicator lines with **line**, **plus/minus**, and **restrict** icons.

It represents the exact position where the node is to be dropped as sibling or child.

The following table explains the usage of indicator icons.

Icons	Description
----- -----	
Plus icon	Indicates that the dragged node is to be added as child of target node.
Minus or restrict icon	Indicates that the dragged node is not to be dropped at the hovered region.
In between icon	Indicates that the dragged node is to be added as siblings of hovered region.

- If you need to prevent dragging action for a particular node, the [nodeDragStart](#) event can be used which is triggered when the node drag is started. If you need to prevent dropping action for a particular node, the [nodeDragStop](#) event can be used which is triggered when the drag is stopped.
- The [nodeDragging](#) event is triggered when the TreeView node is being dragged. You can customize the cloned element in this event.
- The [nodeDropped](#) event is triggered when the TreeView node is dropped on the target element successfully.

In the following sample, the [allowDragAndDrop](#) property is enabled.

APP.COMPONENT.TS

```

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

```

```

import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild } from '@angular/core';
import { TreeViewComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  // specifies the template string for the TreeView component with Drag and Drop
  template: `<div id='treeparent'><ejs-treeview id='treeelement'
[fields]='field' [allowMultiSelection]='allowMultiSelection'
[allowDragAndDrop]='allowDragAndDrop'></ejs-treeview></div>`
})
export class AppComponent {
  @ViewChild('samples')
  public tree?: TreeViewComponent;
  allowMultiSelection: any;
  constructor() {
  }
  // defined the array of data
  public productTeam: Object[] = [
    {
      id: 1, name: 'ASP.NET MVC Team', expanded: true,
      child: [
        { id: 2, pid: 1, name: 'Smith' },
        { id: 3, pid: 1, name: 'Johnson', isSelected: true },
        { id: 4, pid: 1, name: 'Anderson' },
      ]
    },
    {
      id: 5, name: 'Windows Team',
      child: [
        { id: 6, pid: 5, name: 'Clark' },
        { id: 7, pid: 5, name: 'Wright' },
        { id: 8, pid: 5, name: 'Lopez' },
      ]
    },
    {
      id: 9, name: 'Web Team',
      child: [
        { id: 11, pid: 9, name: 'Joshua' },
        { id: 12, pid: 9, name: 'Matthew' },
        { id: 13, pid: 9, name: 'David' },
      ]
    },
    {
      id: 14, name: 'Build Team',
      child: [
        { id: 15, pid: 14, name: 'Ryan' },
        { id: 16, pid: 14, name: 'Justin' },
        { id: 17, pid: 14, name: 'Robert' },
      ]
    },
    {
      id: 18, name: 'WPF Team',
    }
  ]
}

```

```

        child: [
            { id: 19, pid: 18, name: 'Brown' },
            { id: 20, pid: 18, name: 'Johnson' },
            { id: 21, pid: 18, name: 'Miller' },
        ]
    }
};
// maps the appropriate column to fields property
public field: Object = { dataSource: this.productTeam, id: 'id',
parentID: 'pid', text: 'name', hasChildren: 'hasChild', selected:
'isSelected' };
public allowDragAndDrop : 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));

```

Multiple-node drag and drop

To drag and drop more than one node, you should enable the [allowMultiSelection](#) property along with the `allowDragAndDrop` property.

To perform multi-selection, press and hold **CTRL** key and click the desired nodes. To select range of nodes, press and hold the **SHIFT** key and click the nodes.

In the following sample, the `allowMultiSelection` property is enabled along with the `allowDragAndDrop` property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild } from '@angular/core';
import { TreeViewComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  // specifies the template string for the TreeView component with Drag and Drop
  template: `<div id='treeparent'><ejs-treeview id='treeelement'
[fields]='field' [allowMultiSelection]='allowMultiSelection'
[allowDragAndDrop]='allowDragAndDrop'></ejs-treeview></div>`
})
export class AppComponent {
  @ViewChild('samples')
  public tree?: TreeViewComponent;
  constructor() {

```

```

    }
    // defined the array of data
    public productTeam: Object[] = [
        {
            id: 1, name: 'ASP.NET MVC Team', expanded: true,
            child: [
                { id: 2, pid: 1, name: 'Smith' },
                { id: 3, pid: 1, name: 'Johnson', isSelected: true },
                { id: 4, pid: 1, name: 'Anderson', isSelected: true },
            ]
        },
        {
            id: 5, name: 'Windows Team',
            child: [
                { id: 6, pid: 5, name: 'Clark' },
                { id: 7, pid: 5, name: 'Wright' },
                { id: 8, pid: 5, name: 'Lopez' },
            ]
        },
        {
            id: 9, name: 'Web Team',
            child: [
                { id: 11, pid: 9, name: 'Joshua' },
                { id: 12, pid: 9, name: 'Matthew' },
                { id: 13, pid: 9, name: 'David' },
            ]
        },
        {
            id: 14, name: 'Build Team',
            child: [
                { id: 15, pid: 14, name: 'Ryan' },
                { id: 16, pid: 14, name: 'Justin' },
                { id: 17, pid: 14, name: 'Robert' },
            ]
        },
        {
            id: 18, name: 'WPF Team',
            child: [
                { id: 19, pid: 18, name: 'Brown' },
                { id: 20, pid: 18, name: 'Johnson' },
                { id: 21, pid: 18, name: 'Miller' },
            ]
        }
    ];
    // maps the appropriate column to fields property
    public field: Object = { dataSource: this.productTeam, id: 'id',
        parentID: 'pid', text: 'name', hasChildren: 'hasChild', selected:
        'isSelected' };
    // set the Multi Selection option to TreeView
    public allowMultiSelection: boolean = true;
    public allowDragAndDrop : 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));
```

See Also

- [How to restrict the drag-and-drop for particular tree nodes](#)

Template in Angular Treeview component

The TreeView component allows you to customize the look of TreeView nodes by using the [nodeTemplate](#) property. This property accepts either template string or HTML element ID.

In the following sample, employee information such as employee photo, name, and designation have been included using the `nodeTemplate` property.

The template expression should be provided inside the `${...}` interpolation syntax.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  // specifies the template url path
  templateUrl: './template.html',
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  constructor() {
  }
  // defined the array of data
  public localData: Object[] = [
    { id: 1, name: 'Steven Buchanan', eimg: '10', job: 'CEO', hasChild:
true, expanded: true },
    { id: 2, pid: 1, name: 'Laura Callahan', eimg: '2', job: 'Product
Manager', hasChild: true },
    { id: 3, pid: 2, name: 'Andrew Fuller', eimg: '7', job: 'Team Lead',
hasChild: true },
    { id: 4, pid: 3, name: 'Anne Dodsworth', eimg: '1', job: 'Developer'
},
    { id: 5, pid: 1, name: 'Nancy Davolio', eimg: '4', job: 'Product
Manager', hasChild: true },
    { id: 6, pid: 5, name: 'Michael Suyama', eimg: '9', job: 'Team Lead',
hasChild: true },
    { id: 7, pid: 6, name: 'Robert King', eimg: '8', job: 'Developer' },
    { id: 8, pid: 7, name: 'Margaret Peacock', eimg: '6', job:
'Developer' },
    { id: 9, pid: 1, name: 'Janet Leverling', eimg: '3', job: 'HR' },
```

```

    ];
    public field:Object = { dataSource: this.localData, id: 'id', parentID:
    'pid', text: 'name', hasChildren: 'hasChild' };
    public cssClass:string = "custom";
  }

```

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 expand and collapse icons](#)
- [How to customize the tree nodes based on levels](#)

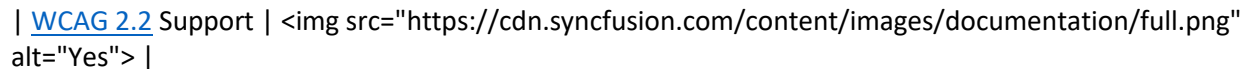
Accessibility in Angular TreeView component

The TreeView 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 TreeView component is outlined below.

| Accessibility Criteria | Compatibility |

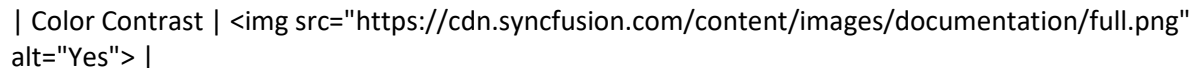
| -- | -- |

| [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 TreeView component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the TreeView component:

Attributes	Purpose
---	---
<code>role=tree</code>	All tree nodes are contained within the element.
<code>role=treeitem</code>	Specifies the role of each tree node in a selectable TreeView and its containment within the tree.
<code>role=group</code>	Specifies the role of each parent node container.
<code>role=checkbox</code>	Indicates checkbox control along with treeitem element.
<code>aria-multiselectable</code>	Indicates whether the TreeView enables multiple selection or not.
<code>aria-expanded</code>	Indicates whether the parent node has expanded or not.
<code>aria-selected</code>	Indicates the selected node.
<code>aria-grabbed</code>	Indicates the selected state on drag-and-drop of node.
<code>aria-level</code>	Indicates the level of node in TreeView.
<code>aria-checked</code>	Indicates the current checked state of TreeView checkbox.
<code>aria-label</code>	Indicates the contextual message for the TreeView checkbox.
<code>aria-activedescendant</code>	Identifies the currently active element when focusing on the TreeView.
<code>aria-disabled</code>	Indicates element is perceivable but disabled.

Keyboard interaction

The TreeView 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 TreeView component.

Interaction Keys	Description
------------------	-------------

|-----|-----|

| Arrow Up | Goes to the previous node. |

| Arrow Down | Goes to the next node. |

| Arrow Right | Expands the current node. |

| Arrow Left | Collapses the current node. |

| Home | Goes to the first node. |

| End | Goes to the last node. |

| F2 | Edits the focused node. |

| Esc | Focuses out the edit state without saving the edited text. |

| Enter | Selects the focused node/saves the edited text. |

| Space | Checks the current node. |

| Ctrl + A | Selects all nodes. |

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion Angular components](#)

How To

Customize the expand and collapse icons in Angular Treeview component

You can customize TreeView expand and collapse icons by using the `cssClass` property of TreeView.

Refer to the sample to customize expand/collapse icons.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild } from '@angular/core';
import { TreeViewComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
```

```

    // specifies the template string for the TreeView component
    template: `<div id='treeparent'><ejs-treeview id='treeelement'
[fields]='field' [cssClass]='cssClass'></ejs-treeview></div>`
  })
  export class AppComponent {
    @ViewChild('samples')
    public tree?: TreeViewComponent;
    constructor() {
    }
    // defined the array of data
    public hierarchicalData: Object[] = [
      { id: '01', name: 'Local Disk(C:)', expanded: true,
        subChild: [
          {
            id: '01-01', name: 'Program Files',
            subChild: [
              { id: '01-01-01', name: '7-Zip' },
              { id: '01-01-02', name: 'Git' },
              { id: '01-01-03', name: 'IIS Express' },
            ]
          },
          {
            id: '01-02', name: 'Users', expanded: true,
            subChild: [
              { id: '01-02-01', name: 'Smith' },
              { id: '01-02-02', name: 'Public' },
              { id: '01-02-03', name: 'Admin' },
            ]
          },
          {
            id: '01-03', name: 'Windows',
            subChild: [
              { id: '01-03-01', name: 'Boot' },
              { id: '01-03-02', name: 'FileManager' },
              { id: '01-03-03', name: 'System32' },
            ]
          },
        ]
      },
      {
        id: '02', name: 'Local Disk(D:)',
        subChild: [
          {
            id: '02-01', name: 'Personals',
            subChild: [
              { id: '02-01-01', name: 'My photo.png' },
              { id: '02-01-02', name: 'Rental document.docx' },
              { id: '02-01-03', name: 'Pay slip.pdf' },
            ]
          },
          {
            id: '02-02', name: 'Projects',
            subChild: [
              { id: '02-02-01', name: 'ASP Application' },
              { id: '02-02-02', name: 'TypeScript Application' },
              { id: '02-02-03', name: 'React Application' },
            ]
          },
        ]
      }
    ]
  }
}

```

```

    },
    {
      id: '02-03', name: 'Office',
      subChild: [
        { id: '02-03-01', name: 'Work details.docx' },
        { id: '02-03-02', name: 'Weekly report.docx' },
        { id: '02-03-03', name: 'Wish list.csv' },
      ]
    },
  ],
},
{
  id: '03', name: 'Local Disk(E:)', icon: 'folder',
  subChild: [
    {
      id: '03-01', name: 'Pictures',
      subChild: [
        { id: '03-01-01', name: 'Wind.jpg' },
        { id: '03-01-02', name: 'Stone.jpg' },
        { id: '03-01-03', name: 'Home.jpg' },
      ]
    },
    {
      id: '03-02', name: 'Documents',
      subChild: [
        { id: '03-02-01', name: 'Environment Pollution.docx' },
        { id: '03-02-02', name: 'Global Warming.ppt' },
        { id: '03-02-03', name: 'Social Network.pdf' },
      ]
    },
    {
      id: '03-03', name: 'Study Materials',
      subChild: [
        { id: '03-03-01', name: 'UI-Guide.pdf' },
        { id: '03-03-02', name: 'Tutorials.zip' },
        { id: '03-03-03', name: 'TypeScript.7z' },
      ]
    },
  ],
},
];
// maps the appropriate column to fields property
public field: Object = { dataSource: this.hierarchicalData, id: 'id',
text: 'name', child: 'subChild' };
// set the Multi Selection option to TreeView
public cssClass: string = "custom";
}

```

MAIN.TS

```

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

```

Process the tree node operations using context menu in Angular Treeview component

You can intergrade the context menu with 'TreeView' component in order to perform the tree view related operations like add, remove and renaming node.

Following is an example which demonstrates the above cases which are used to manipulate tree view operations in the 'select' event of context menu.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule, BeforeOpenCloseMenuEventArgs, MenuEventArgs,
MenuItemModel, ContextMenuComponent, ContextMenuModule } from
'@syncfusion/ej2-angular-navigations'
import { Component, Inject, ViewChild } from '@angular/core';
import { TreeViewComponent, NodeClickEventArgs, BeforeOpenCloseMenuEventArgs,
MenuEventArgs, MenuItemModel, ContextMenuComponent } from '@syncfusion/ej2-
angular-navigations';
@Component({
imports: [
FormsModule, TreeViewModule, ContextMenuModule
],
standalone: true,
selector: 'app-container',
template: `<div id='treeparent'>
<ejs-treeview id='tree' #treevalidate [fields]='field'
(nodeClicked)='nodeclicked($event)'></ejs-treeview>
</div>
<ejs-contextmenu #contentmenutree id='contentmenutree'
target='#tree' [items]='menuItems' (beforeOpen)='beforeopen($event)'
(select)='menuclick($event)'></ejs-contextmenu>`
})
export class AppComponent {
public hierarchicalData: Object[] = [
{ id: '01', name: 'Local Disk (C:)', expanded: true,
hasAttribute:{class:'remove rename'},
subChild: [
{
id: '01-01', name: 'Program Files',
subChild: [
{ id: '01-01-01', name: 'Windows NT' },
{ id: '01-01-02', name: 'Windows Mail' },
{ id: '01-01-03', name: 'Windows Photo Viewer' },
]
},
{
id: '01-02', name: 'Users', expanded: true,
subChild: [
{ id: '01-02-01', name: 'Smith' },
{ id: '01-02-02', name: 'Public' },
{ id: '01-02-03', name: 'Admin' },
]
},
{
id: '01-03', name: 'Windows',
subChild: [
```

```

{ id: '01-03-01', name: 'Boot' },
{ id: '01-03-02', name: 'FileManager' },
{ id: '01-03-03', name: 'System32' },
]
},
]
},
{
  id: '02', name: 'Local Disk (D:)', hasAttribute:{class:'remove'},
  subChild: [
    {
      id: '02-01', name: 'Personals',
      subChild: [
        { id: '02-01-01', name: 'My photo.png' },
        { id: '02-01-02', name: 'Rental document.docx' },
        { id: '02-01-03', name: 'Pay slip.pdf' },
      ]
    },
    {
      id: '02-02', name: 'Projects',
      subChild: [
        { id: '02-02-01', name: 'ASP Application' },
        { id: '02-02-02', name: 'TypeScript Application' },
        { id: '02-02-03', name: 'React Application' },
      ]
    },
    {
      id: '02-03', name: 'Office',
      subChild: [
        { id: '02-03-01', name: 'Work details.docx' },
        { id: '02-03-02', name: 'Weekly report.docx' },
        { id: '02-03-03', name: 'Wish list.csv' },
      ]
    },
  ],
},
],
{
  id: '03', name: 'Local Disk (E:)', icon: 'folder',
  hasAttribute:{class:'rename'},
  subChild: [
    {
      id: '03-01', name: 'Pictures',
      subChild: [
        { id: '03-01-01', name: 'Wind.jpg' },
        { id: '03-01-02', name: 'Stone.jpg' },
        { id: '03-01-03', name: 'Home.jpg' },
      ]
    },
    {
      id: '03-02', name: 'Documents',
      subChild: [
        { id: '03-02-01', name: 'Environment Pollution.docx' },
        { id: '03-02-02', name: 'Global Warming.ppt' },
        { id: '03-02-03', name: 'Social Network.pdf' },
      ]
    },
  ],
},
],
}

```

```

        {
            id: '03-03', name: 'Study Materials',
            subChild: [
                { id: '03-03-01', name: 'UI-Guide.pdf' },
                { id: '03-03-02', name: 'Tutorials.zip' },
                { id: '03-03-03', name: 'TypeScript.7z' },
            ]
        },
    ],
}

];
// Mapping TreeView fields property with data source properties
public field:Object={ dataSource: this.hierarchicalData, id: 'id', text:
'name', child: 'subChild', htmlAttributes: 'hasAttribute' };
@ViewChild('treevalidate') treevalidate?: TreeViewComponent;
@ViewChild('contentmenutree') contentmenutree?: ContextMenuComponent;
public nodeclicked(args: NodeClickEventArgs) {
    if (args.event.which === 3) {
        (this.treevalidate as TreeViewComponent).selectedNodes =
[args.node.getAttribute('data-uid') as string];
    }
}

//Render the context menu with target as Treeview
public menuItems: MenuItemModel[] = [
    { text: 'Add New Item' },
    { text: 'Rename Item' },
    { text: 'Remove Item' }
];
public index: number = 1;
public menuclick(args: MenuEventArgs) {
    let targetNodeId: string = this.treevalidate?.selectedNodes[0] as string;
    if (args.item.text == "Add New Item") {
        let nodeId: string = "tree_" + this.index;
        let item: { [key: string]: Object } = { id: nodeId, name: "New Folder" };
        this.treevalidate?.addNodes([item], targetNodeId, null as any);
        this.index++;
        this.hierarchicalData.push(item);
        this.treevalidate?.beginEdit(nodeId);
    }
    else if (args.item.text == "Remove Item") {
        this.treevalidate?.removeNodes([targetNodeId]);
    }
    else if (args.item.text == "Rename Item") {
        this.treevalidate?.beginEdit(targetNodeId);
    }
}
public beforeopen(args: BeforeOpenCloseMenuEventArgs) {
    let targetNodeId: string = this.treevalidate?.selectedNodes[0] as string;
    let targetNode: Element = document.querySelector('[data-uid="' +
targetNodeId + '"]') as Element;
    if (targetNode.classList.contains('remove')) {
        this.contentmenutree?.enableItems(['Remove Item'], false);
    }
    else {
        this.contentmenutree?.enableItems(['Remove Item'], true);
    }
    if (targetNode.classList.contains('rename')) {

```

```

        this.contentmenutree?.enableItems(['Rename Item'], false);
    }
    else {
        this.contentmenutree?.enableItems(['Rename Item'], true);
    }
}
}

```

MAIN.TS

```

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

```

Check uncheck the checkbox on clicking the tree node text in Angular Treeview component

You can check and uncheck the checkboxes of tree view by clicking the tree node using the **nodeClicked** event of TreeView.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, Inject, ViewChild } from '@angular/core';
import { TreeViewComponent, NodeKeyPressEventArgs, NodeClickEventArgs } from '@syncfusion/ej2-angular-navigations';
/**
 * TreeView Checkboxes sample
 */
@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div id='treeparent'><ejs-treeview id='treeElement'
#treevalidate [fields]='field' [showCheckBox]='showCheckBox'
(nodeClicked)='nodeCheck($event)' (keyPress)='nodeCheck($event)'></ejs-
treeview></div>`
})
export class AppComponent {
  // Data source for TreeView component
  public countries: Object[] = [
    { id: 1, name: 'Australia', hasChild: true, expanded: true },
    { id: 2, pid: 1, name: 'New South Wales' },
    { id: 3, pid: 1, name: 'Victoria' },
    { id: 4, pid: 1, name: 'South Australia' },
    { id: 6, pid: 1, name: 'Western Australia' },
    { id: 7, name: 'Brazil', hasChild: true },
    { id: 8, pid: 7, name: 'Paraná' },
    { id: 9, pid: 7, name: 'Ceará' },
    { id: 10, pid: 7, name: 'Acre' },
    { id: 11, name: 'China', hasChild: true },

```



```

    { id: 12, pid: 11, name: 'Guangzhou' },
    { id: 13, pid: 11, name: 'Shanghai' },
    { id: 14, pid: 11, name: 'Beijing' },
    { id: 15, pid: 11, name: 'Shantou' },
    { id: 16, name: 'France', hasChild: true },
    { id: 17, pid: 16, name: 'Pays de la Loire' },
    { id: 18, pid: 16, name: 'Aquitaine' },
    { id: 19, pid: 16, name: 'Brittany' },
    { id: 20, pid: 16, name: 'Lorraine' },
    { id: 21, name: 'India', hasChild: true },
    { id: 22, pid: 21, name: 'Assam' },
    { id: 23, pid: 21, name: 'Bihar' },
    { id: 24, pid: 21, name: 'Tamil Nadu' },
    { id: 25, pid: 21, name: 'Punjab' }
  ];
  public field:Object = { dataSource: this.countries, id: 'id', parentID:
    'pid', text: 'name', hasChildren: 'hasChild' };
  // Enable the checkbox for TreeView
  public showCheckBox:boolean = true;
  @ViewChild('treevalidate') treevalidate?: TreeViewComponent;
  public nodeCheck(args: NodeKeyPressEventArgs | NodeClickEventArgs | any):
void {
    let checkedNode: any = [args.node];
    if ((args.event.target as EventTarget | any).classList.contains('e-
fullrow') || args.event.key == "Enter") {
      let getNodeDetails: any = (this.treevalidate as TreeViewComponent
).getNode(args.node);
      if (getNodeDetails.isChecked == 'true') {
        (this.treevalidate as TreeViewComponent).uncheckAll(checkedNode);
      } else {
        (this.treevalidate as TreeViewComponent).checkAll(checkedNode);
      }
    }
  }
}

```

MAIN.TS

```

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

```

Validate the text when renaming the tree node in Angular Treeview component

You can validate the tree node text while editing using `nodeEdited` event of the TreeView.

Following is an example that shows how to validate and prevent empty values in tree node.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild } from '@angular/core';

```

```

import { TreeViewComponent, NodeEditEventArgs } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  // specifies the template string for the TreeView component
  template: `<div id='treeparent'><ejs-treeview id='treeElement'
#treevalidate [fields]='field' [allowEditing]='allowEditing'
(nodeEdited)='onNodeEdited($event)'></ejs-treeview></div>
<div id="display"></div>`
})
export class AppComponent {
  @ViewChild('samples')
  public tree?: TreeViewComponent;
  constructor() {
    // Hierarchical data source for TreeView component
    public hierarchicalData: Object[] = [
      {
        id: 1, name: 'Discover Music', expanded: true,
        child: [
          { id: 2, name: 'Hot Singles' },
          { id: 3, name: 'Rising Artists' },
          { id: 4, name: 'Live Music' }
        ]
      },
      {
        id: 7, name: 'Sales and Events',
        child: [
          { id: 8, name: '100 Albums - $5 Each' },
          { id: 9, name: 'Hip-Hop and R&B Sale' },
          { id: 10, name: 'CD Deals' }
        ]
      },
      {
        id: 11, name: 'Categories',
        child: [
          { id: 12, name: 'Songs' },
          { id: 13, name: 'Bestselling Albums' },
          { id: 14, name: 'New Releases' },
          { id: 15, name: 'Bestselling Songs' }
        ]
      },
      {
        id: 16, name: 'MP3 Albums',
        child: [
          { id: 17, name: 'Rock' },
          { id: 18, name: 'Gospel' },
          { id: 19, name: 'Latin Music' },
          { id: 20, name: 'Jazz' }
        ]
      },
      {
        id: 21, name: 'More in Music',

```

```

        child: [
          { id: 22, name: 'Music Trade-In' },
          { id: 23, name: 'Redeem a Gift Card' },
          { id: 24, name: 'Band T-Shirts' }
        ]
      }
    ];
    // Mapping TreeView fields property with data source properties
    public field:Object={ dataSource: this.hierarchicalData, id: 'id', text:
    'name', child: 'child' };
    public allowEditing: boolean = true;
    @ViewChild ('treevalidate') treevalidate?: TreeViewComponent;
    public onNodeEdited(args: NodeEditEventArgs): void {
      let displayContent:string = "";
      if (args.newText.trim() == "") {
        args.cancel=true;
        displayContent = "TreeView item text should not be empty";
      } else if (args.newText != args.oldText) {
        displayContent = "TreeView item text edited successfully";
      } else {
        displayContent = "";
      }
      (document.getElementById("display") as
      HTMLElement).innerHTML = displayContent;
    }
  }
}

```

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 tree nodes based on levels in Angular Treeview component

You can customize the tree nodes level wise by adding custom cssClass to the component and enabling styles.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule,TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div id='treeparent'><ejs-treeview id='treeElement'
#treevalidate [fields]='field' [cssClass]='cssClass'></ejs-treeview></div>
<div class="details">
<label>Note:</label>

```

```

        <div><b>1. The font-weight "Bold" is applied for all
the leaf nodes</b></div>
        <div><i>2. The font-weight "Italic" is applied for
first level nodes</i></div>
        <div style="color: darkmagenta">3. The color
"darkmagenta" is applied for second level nodes</div>
    </div>`
))
export class AppComponent {
    // Hierarchical data source for TreeView component
    public hierarchicalData: Object[] = [
        { id: '01', name: 'Local Disk (C:)', expanded: true,
          subChild: [
            {
              id: '01-01', name: 'Program Files',
              subChild: [
                { id: '01-01-01', name: 'Windows NT' },
                { id: '01-01-02', name: 'Windows Mail' },
                { id: '01-01-03', name: 'Windows Photo Viewer' },
              ]
            },
            {
              id: '01-02', name: 'Users', expanded: true,
              subChild: [
                { id: '01-02-01', name: 'Smith' },
                { id: '01-02-02', name: 'Public' },
                { id: '01-02-03', name: 'Admin' },
              ]
            },
            {
              id: '01-03', name: 'Windows',
              subChild: [
                { id: '01-03-01', name: 'Boot' },
                { id: '01-03-02', name: 'FileManager' },
                { id: '01-03-03', name: 'System32' },
              ]
            },
          ]
        },
        {
          id: '02', name: 'Local Disk (D:)',
          subChild: [
            {
              id: '02-01', name: 'Personals',
              subChild: [
                { id: '02-01-01', name: 'My photo.png' },
                { id: '02-01-02', name: 'Rental document.docx' },
                { id: '02-01-03', name: 'Pay slip.pdf' },
              ]
            },
            {
              id: '02-02', name: 'Projects',
              subChild: [
                { id: '02-02-01', name: 'ASP Application' },
                { id: '02-02-02', name: 'TypeScript Application' },
                { id: '02-02-03', name: 'React Application' },
              ]
            },
          ]
        }
      ]
    }
  }

```

```

    },
    {
      id: '02-03', name: 'Office',
      subChild: [
        { id: '02-03-01', name: 'Work details.docx' },
        { id: '02-03-02', name: 'Weekly report.docx' },
        { id: '02-03-03', name: 'Wish list.csv' },
      ]
    },
  ],
},
{
  id: '03', name: 'Local Disk (E:)', icon: 'folder',
  subChild: [
    {
      id: '03-01', name: 'Pictures',
      subChild: [
        { id: '03-01-01', name: 'Wind.jpg' },
        { id: '03-01-02', name: 'Stone.jpg' },
        { id: '03-01-03', name: 'Home.jpg' },
      ]
    },
    {
      id: '03-02', name: 'Documents',
      subChild: [
        { id: '03-02-01', name: 'Environment Pollution.docx' },
        { id: '03-02-02', name: 'Global Warming.ppt' },
        { id: '03-02-03', name: 'Social Network.pdf' },
      ]
    },
    {
      id: '03-03', name: 'Study Materials',
      subChild: [
        { id: '03-03-01', name: 'UI-Guide.pdf' },
        { id: '03-03-02', name: 'Tutorials.zip' },
        { id: '03-03-03', name: 'TypeScript.7z' },
      ]
    },
  ],
}
];
// Mapping TreeView fields property with data source properties
public field:Object = { dataSource: this.hierarchicalData, id: 'id', text:
'name', child: 'subChild' };
public cssClass: string = 'mytree';
}

```

MAIN.TS

```

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

```

Restrict the drag and drop for particular tree nodes in Angular Treeview component

You can able to restrict to drag and drop files under folder only.

These can be achieved by using 'nodeDragStop' and 'nodeDragging' event of TreeView.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, Inject, ViewChild } from '@angular/core';
import { DragAndDropEventArgs } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div id='treeparent'><ejs-treeview id="icons"
[fields]='field' sortOrder='Ascending' allowDragAndDrop='allowDragAndDrop'
(nodeDragStop)='dragStop($event)' (nodeDragging)='nodeDrag($event)'></ejs-
treeview></div>`
})
export class AppComponent {
  // Hierarchical data source for TreeView component
  public hierarchicalData: Object[] = [
    {
      nodeId: '01', nodeText: 'Music', icon: 'folder',
      nodeChild: [
        { nodeId: '01-01', nodeText: 'Gouttes.mp3', icon: 'audio' }
      ]
    },
    {
      nodeId: '02', nodeText: 'Videos', icon: 'folder',
      nodeChild: [
        { nodeId: '02-01', nodeText: 'Naturals.mp4', icon: 'video' },
        { nodeId: '02-02', nodeText: 'Wild.mpeg', icon: 'video' },
      ]
    },
    {
      nodeId: '03', nodeText: 'Documents', icon: 'folder',
      nodeChild: [
        { nodeId: '03-01', nodeText: 'Environment Pollution.docx',
          icon: 'docx' },
        { nodeId: '03-02', nodeText: 'Global Water, Sanitation, &
          Hygiene.docx', icon: 'docx' },
        { nodeId: '03-03', nodeText: 'Global Warming.ppt', icon:
          'ppt' },
        { nodeId: '03-04', nodeText: 'Social Network.pdf', icon:
          'pdf' },
        { nodeId: '03-05', nodeText: 'Youth Empowerment.pdf', icon:
          'pdf' },
      ]
    },
  ],
}
```

```

        nodeId: '04', nodeText: 'Pictures', icon: 'folder', expanded:
true,
        nodeChild: [
            {
                nodeId: '04-01', nodeText: 'Camera Roll', icon: 'folder',
expanded: true,
                nodeChild: [
                    { nodeId: '04-01-01', nodeText:
'WIN_20160726_094117.JPG', image:
'https://ej2.syncfusion.com/demos/src/treeview/images/employees/9.png' },
                    { nodeId: '04-01-02', nodeText:
'WIN_20160726_094118.JPG', image:
'https://ej2.syncfusion.com/demos/src/treeview/images/employees/3.png' },
                ]
            },
            { nodeId: '04-02', nodeText: 'Wind.jpg', icon: 'images' },
            { nodeId: '04-03', nodeText: 'Stone.jpg', icon: 'images' },
        ]
    },
    {
        nodeId: '05', nodeText: 'Downloads', icon: 'folder',
        nodeChild: [
            { nodeId: '05-01', nodeText: 'UI-Guide.pdf', icon: 'pdf' },
            { nodeId: '05-02', nodeText: 'Tutorials.zip', icon: 'zip' },
            { nodeId: '05-03', nodeText: 'Game.exe', icon: 'exe' },
            { nodeId: '05-04', nodeText: 'TypeScript.7z', icon: 'zip' },
        ]
    }
];
    public field:Object = { dataSource: this.hierarchicalData, id: 'nodeId',
text: 'nodeText', child: 'nodeChild', iconCss: 'icon', imageUrl: 'image' };
    public allowDragAndDrop: boolean = true;
    public nodeDrag(args: DragAndDropEventArgs): void {
        if (args.droppedNode != null &&
args.droppedNode.getElementsByClassName('folder') &&
args.droppedNode.getElementsByClassName('folder').length === 0) {
            args.dropIndicator = 'e-no-drop';
        }
    }
    public dragStop(args: DragAndDropEventArgs): void {
        if (args.droppedNode != null &&
args.droppedNode.getElementsByClassName('folder') &&
args.droppedNode.getElementsByClassName('folder').length === 0) {
            args.cancel = true;
        }
    }
}

```

MAIN.TS

```

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

```

Accordion tree in Angular Treeview component

Accordion is an interface where a list of items can be collapsed or expanded, but only one list can be collapsed or expanded at a time. You can customize the TreeView to make it behave as an accordion. Refer to the following code sample to create an accordion tree.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, Inject, ViewChild } from '@angular/core';
import { TreeViewComponent, NodeSelectEventArgs } from '@syncfusion/ej2-angular-navigations';
/**
 * TreeView Accordion sample
 */
@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div id='treeparent'><ejs-treeview id='treeElement'
#treevalidate [fields]='field' (nodeSelected)='nodeSelect($event)'
[cssClass]='cssClass'></ejs-treeview></div>`
})
export class AppComponent {
  // Data source for TreeView component
  public continents: Object[] = [
    {
      code: "AF", name: "Africa", countries: [
        { code: "NGA", name: "Nigeria" },
        { code: "EGY", name: "Egypt" },
        { code: "ZAF", name: "South Africa" }
      ]
    },
    {
      code: "AS", name: "Asia", countries: [
        { code: "CHN", name: "China" },
        { code: "IND", name: "India", selected: true },
        { code: "JPN", name: "Japan" }
      ]
    },
    {
      code: "EU", name: "Europe", countries: [
        { code: "DNK", name: "Denmark" },
        { code: "FIN", name: "Finland" },
        { code: "AUT", name: "Austria",
        }
      ]
    },
    {
      code: "NA", name: "North America", countries: [
        { code: "USA", name: "United States of America" },
        { code: "CUB", name: "Cuba" },
        { code: "MEX", name: "Mexico" }
      ]
    }
  ]
}
```



```

    ]
    },
    {
        code: "OC", name: "Oceania", countries: [
            { code: "AUS", name: "Australia" },
            { code: "NZL", name: "New Zealand" },
            { code: "WSM", name: "Samoa" }
        ]
    }
];
public field:Object = { dataSource: this.continents, id: "code", text:
"name", child: "countries" };
public cssClass="accordiontree";
@ViewChild ('treevalidate') tree?: TreeViewComponent;
public nodeSelect(args: NodeSelectEventArgs): void {
    if (args.node.classList.contains('e-level-1')) {
        this.tree?.collapseAll();
        this.tree?.expandAll([args.node]);
        (this.tree as TreeViewComponent).expandOn = 'None';
    }
}
}
}

```

MAIN.TS

```

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

```

Auto hide show expand collapse icon in Angular Treeview component

You can display the expand icon by hovering the mouse over TreeView and hide the expand icon by leaving the mouse from TreeView. Refer to the following code sample to hide/show the expand/collapse icon automatically using the mouse.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, Inject, ViewChild } from '@angular/core';
import { TreeViewComponent } from '@syncfusion/ej2-angular-navigations';
/**
 * TreeView Auto hide/show expand/collapse icons
 */
@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div id='treeparent'><ejs-treeview id='treeElement'
#treevalidate [fields]='field' (created)='onCreate($event)'></ejs-
treeview></div>`

```

```

})
export class AppComponent {
  // Data source for TreeView component
  public countries: Object[] = [
    { id: 1, name: 'India', hasChild: true },
    { id: 2, pid: 1, name: 'Assam' },
    { id: 3, pid: 1, name: 'Bihar' },
    { id: 4, pid: 1, name: 'Tamil Nadu' },
    { id: 6, pid: 1, name: 'Punjab' },
    { id: 7, name: 'Brazil', hasChild: true },
    { id: 8, pid: 7, name: 'Paraná' },
    { id: 9, pid: 7, name: 'Ceará' },
    { id: 10, pid: 7, name: 'Acre' },
    { id: 11, name: 'France', hasChild: true },
    { id: 12, pid: 11, name: 'Pays de la Loire' },
    { id: 13, pid: 11, name: 'Aquitaine' },
    { id: 14, pid: 11, name: 'Brittany' },
    { id: 15, pid: 11, name: 'Lorraine' },
    { id: 16, name: 'Australia', hasChild: true },
    { id: 17, pid: 16, name: 'New South Wales' },
    { id: 18, pid: 16, name: 'Victoria' },
    { id: 19, pid: 16, name: 'South Australia' },
    { id: 20, pid: 16, name: 'Western Australia' },
    { id: 21, name: 'China', hasChild: true },
    { id: 22, pid: 21, name: 'Guangzhou' },
    { id: 23, pid: 21, name: 'Shanghai' },
    { id: 24, pid: 21, name: 'Beijing' },
    { id: 25, pid: 21, name: 'Shantou' }
  ];

  public field:Object = { dataSource: this.countries, id: 'id', text:
    'name', parentID: 'pid', hasChildren: 'hasChild' };
  @ViewChild('treevalidate') tree?: TreeViewComponent;
  public onCreate(args: any): void {
    let collapse: NodeListOf<Element> = (this.tree as
any)?.element.querySelectorAll('.e-icons.e-icon-collapsible') as
NodeListOf<Element>;
    let expand: NodeListOf<Element> = (this.tree as
any)?.element.querySelectorAll('.e-icons.e-icon-expandable') as
NodeListOf<Element>;
    this.hideIcon(expand, collapse);
    (this.tree as any)?.element.addEventListener('mouseenter', (event:any)
=> {
      this.showIcon(expand, collapse);
    });
    (this.tree as any)?.element.addEventListener('mouseleave', (event:any)
=> {
      this.hideIcon(expand, collapse);
    });
  }
  // hides expand/collapse icon on hovering the mouse
  public hideIcon(expand: NodeListOf<Element>, collapse:
NodeListOf<Element>) {
    for(let i: number = 0; i < collapse.length; i++){
      collapse[i].setAttribute('style','visibility: hidden');
    }
    for(let j: number = 0; j < expand.length; j++){
      expand[j].setAttribute('style','visibility: hidden');
    }
  }
}

```

```

    }
  }
  // shows expand/collapse icon while leaving the mouse
  public showIcon(expand: NodeListOf<Element>, collapse: NodeListOf<Element>)
  {
    for(let i: number = 0; i < collapse.length; i++ ){
      collapse[i].setAttribute('style','visibility');
    }
    for(let j: number = 0; j < expand.length; j++ ){
      expand[j].setAttribute('style','visibility');
    }
  }
}

```

MAIN.TS

```

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

```

Filtering tree nodes in Angular Treeview component

You can filter the tree nodes based on their text using the **DataManager** plugin and the **fields** property of the TreeView.

The following code example demonstrates how to filter the tree nodes in a TreeView.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { MaskedTextBoxModule } from '@syncfusion/ej2-angular-inputs'
import { FieldsSettingsModel, TreeViewComponent } from "@syncfusion/ej2-angular-navigations"
import { MaskedTextBoxComponent } from "@syncfusion/ej2-angular-inputs"
import { Component, Inject, ViewChild } from '@angular/core';
import { DataManager, Query, ReturnOption, Predicate } from '@syncfusion/ej2-data';
/**
 * Filtering tree nodes sample
 */
@Component({
  imports: [
    FormsModule, TreeViewModule, MaskedTextBoxModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div id='treeparent'><ejs-maskedtextbox #maskObj id="search"
(change)="searchNodes($event)"></ejs-maskedtextbox><ejs-treeview #treeviewObj
id="default" [fields]='field'></ejs-treeview></div>`
})
export class AppComponent {
  @ViewChild("treeviewObj") listTreeObj?: TreeViewComponent;
  @ViewChild("maskObj") maskObj?: MaskedTextBoxComponent;

```

```

// list data source for TreeView component
public localData: Object[] = [
    { id: 1, name: "Australia", hasChild: true },
    { id: 2, pid: 1, name: "New South Wales" },
    { id: 3, pid: 1, name: "Victoria" },
    { id: 4, pid: 1, name: "South Australia" },
    { id: 6, pid: 1, name: "Western Australia" },
    { id: 7, name: "Brazil", hasChild: true },
    { id: 8, pid: 7, name: "Paraná" },
    { id: 9, pid: 7, name: "Ceará" },
    { id: 10, pid: 7, name: "Acre" },
    { id: 11, name: "China", hasChild: true },
    { id: 12, pid: 11, name: "Guangzhou" },
    { id: 13, pid: 11, name: "Shanghai" },
    { id: 14, pid: 11, name: "Beijing" },
    { id: 15, pid: 11, name: "Shantou" },
    { id: 16, name: "France", hasChild: true },
    { id: 17, pid: 16, name: "Pays de la Loire" },
    { id: 18, pid: 16, name: "Aquitaine" },
    { id: 19, pid: 16, name: "Brittany" },
    { id: 20, pid: 16, name: "Lorraine" },
    { id: 21, name: "India", hasChild: true },
    { id: 22, pid: 21, name: "Assam" },
    { id: 23, pid: 21, name: "Bihar" },
    { id: 24, pid: 21, name: "Tamil Nadu" },
    { id: 25, pid: 21, name: "Punjab" }
];

// Mapping TreeView fields property with data source properties
public field:Object = { dataSource: this.localData, id: 'id', parentID:
'pid', text: 'name', hasChildren: 'hasChild', expanded: "expanded" }
//Change the dataSource for TreeView
public changeDataSource(data: Object[]) {
    (this.listTreeObj as TreeViewComponent).fields = {
        dataSource: data, id: 'id', text: 'name', parentID: 'pid',
hasChildren: 'hasChild'
    } as FieldsSettingsModel;
}

//Filtering the TreeNodes
public searchNodes(args: any) {
    let _text = this.maskObj?.element.value;
    let predicates = [], _array = [], _filter = [];
    if (_text == "") {
        this.changeDataSource(this.localData);
    }
    else {
        let predicate = new Predicate('name', 'startswith', _text as
string, true);
        let filteredList = new
DataManager(this.localData).executeLocal(new Query().where(predicate));
        console.log(filteredList)
        for (let j = 0; j < filteredList.length; j++) {
            _filter.push((filteredList as any)[j]["id"]);
            let filters = this.getFilterItems(filteredList[j],
this.localData);
            for (let i = 0; i < filters.length; i++) {
                if (_array.indexOf(filters[i]) == -1 && filters[i] !=
null) {

```

```

        _array.push(filters[i]);
        predicates.push(new Predicate('id', 'equal',
filters[i] as any, false));
    }
}
}
if (predicates.length == 0) {
    this.changeDataSource([]);
} else {
    let query = new Query().where(Predicate.or(predicates));
    let newList = new
DataManager(this.localData).executeLocal(query);
    this.changeDataSource(newList);
    let proxy = this;
    setTimeout(function (this: any) {
        proxy.listTreeObj?.expandAll();
    }, 100);
}
}
}
//Find the Parent Nodes for corresponding childs
public getFilterItems(fList: Object | any, list: Object[]): Object[] {
    let nodes = [];
    nodes.push(fList["id"] as any);
    let query2 = new Query().where('id', 'equal', fList["pid"], false);
    let fList1 = new DataManager(list).executeLocal(query2);
    if (fList1.length != 0) {
        let pNode = this.getFilterItems(fList1[0], list);
        for (let i = 0; i < pNode.length; i++) {
            if (nodes.indexOf(pNode[i]) == -1 && pNode[i] != null)
                nodes.push(pNode[i]);
        }
        return nodes;
    }
    return nodes;
}
}
}
}

```

MAIN.TS

```

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

```

Set tool tip for tree nodes in Angular Treeview component

TreeView control allows you to set tooltip option to tree nodes using the [tooltip](#) property. The following code example demonstrates how to set tooltip for TreeView nodes.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'

```

```

import { Component, Inject, ViewChild } from '@angular/core';
import { TreeViewComponent } from '@syncfusion/ej2-angular-navigations';
/**
 * TreeView tooltip sample
 */
@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div id='treeparent'><ejs-treeview [fields]='field'></ejs-treeview></div>`
})
export class AppComponent {
  // Data source for TreeView component
  public hierarchicalData: Object[] = [
    { id: '01', name: 'Local Disk (C:)', expanded: true, tooltip: 'Local Disk (C:)',
      subChild: [
        {
          id: '01-01', name: 'Program Files', tooltip: 'Program Files',
          subChild: [
            { id: '01-01-01', name: 'Windows NT', tooltip: 'Windows NT' },
            { id: '01-01-02', name: 'Windows Mail', tooltip: 'Windows Mail' },
            { id: '01-01-03', name: 'Windows Photo Viewer', tooltip: 'Windows Photo Viewer' },
          ]
        },
        {
          id: '01-02', name: 'Users', expanded: true, tooltip: 'Users',
          subChild: [
            { id: '01-02-01', name: 'Smith', tooltip: 'Smith' },
            { id: '01-02-02', name: 'Public', tooltip: 'Public' },
            { id: '01-02-03', name: 'Admin', tooltip: 'Admin' },
          ]
        },
        {
          id: '01-03', name: 'Windows', tooltip: 'Windows',
          subChild: [
            { id: '01-03-01', name: 'Boot', tooltip: 'Boot' },
            { id: '01-03-02', name: 'FileManager', tooltip: 'FileManager' },
            { id: '01-03-03', name: 'System32', tooltip: 'System32' },
          ]
        }
      ]
    },
    {
      id: '02', name: 'Local Disk (D:)', tooltip: 'Local Disk (D:)',
      subChild: [
        {
          id: '02-01', name: 'Personals', tooltip: 'Personals',
          subChild: [

```

```

        { id: '02-01-01', name: 'My photo.png', tooltip: 'My
photo.png' },
        { id: '02-01-02', name: 'Rental document.docx', tooltip:
'Rental document.docx' },
        { id: '02-01-03', name: 'Pay slip.pdf', tooltip: 'Pay
slip.pdf' },
    ],
},
{
    id: '02-02', name: 'Projects', tooltip: 'Projects',
    subChild: [
        { id: '02-02-01', name: 'ASP Application', tooltip: 'ASP
Application' },
        { id: '02-02-02', name: 'TypeScript Application',
tooltip: 'TypeScript Application' },
        { id: '02-02-03', name: 'React Application' , tooltip:
'React Application'},
    ]
},
{
    id: '02-03', name: 'Office', tooltip: 'Office',
    subChild: [
        { id: '02-03-01', name: 'Work details.docx' ,
tooltip: 'Work details.docx' },
        { id: '02-03-02', name: 'Weekly report.docx', tooltip:
'Weekly report.docx' },
        { id: '02-03-03', name: 'Wish list.csv', tooltip: 'Wish
list.csv' },
    ]
},
],
},
{
    id: '03', name: 'Local Disk (E:)', tooltip: 'Local Disk (E:)',
    subChild: [
        {
            id: '03-01', name: 'Pictures', tooltip: 'Pictures',
            subChild: [
                { id: '03-01-01', name: 'Wind.jpg', tooltip: 'Wind.jpg'
},
                { id: '03-01-02', name: 'Stone.jpg', tooltip: 'Stone.jpg'
},
                { id: '03-01-03', name: 'Home.jpg', tooltip: 'Home.jpg'
},
            ]
        },
        {
            id: '03-02', name: 'Documents', tooltip: 'Documents',
            subChild: [
                { id: '03-02-01', name: 'Environment Pollution.docx' ,
tooltip: 'Environment Pollution.docx' },
                { id: '03-02-02', name: 'Global Warming.ppt', tooltip:
'Global Warming.ppt' },
                { id: '03-02-03', name: 'Social Network.pdf', tooltip:
'Social Network.pdf' },
            ]
        },
    ],
},

```

```

        {
            id: '03-03', name: 'Study Materials', tooltip: 'Study
Materials',
            subChild: [
                { id: '03-03-01', name: 'UI-Guide.pdf' , tooltip: 'UI-
Guide.pdf' },
                { id: '03-03-02', name: 'Tutorials.zip', tooltip:
'Tutorials.zip' },
                { id: '03-03-03', name: 'TypeScript.7z' , tooltip:
'TypeScript.7z' },
            ]
        },
    ],
    },
    ];
    public field:Object = {  dataSource: this.hierarchicalData, id: 'id',
text: 'name', child: 'subChild' };
}

```

MAIN.TS

```

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

```

Sorting treeview level wise in Angular Treeview component

You can sort the TreeView nodes based on their level. When using the `sortOrder` property, the whole TreeView is sorted. When you sort a particular level, you can use the following code sample. The following code sample demonstrates how to sort the parent node alone in TreeView.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, Inject, ViewChild } from '@angular/core';
import { TreeViewComponent, NodeExpandEventArgs, FieldsSettingsModel } from '@syncfusion/ej2-angular-navigations';
import { DataManager, Query } from '@syncfusion/ej2-data';
/**
 * Treeview Disable check box of parent nodes sample
 */
@Component({
  imports: [
    FormsModule,TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div id='treeparent'><ejs-treeview id='treeElement'
#treevalidate [fields]='field' (nodeExpanding)='onNodeExpand($event)'
(created)='onCreate($event)'></ejs-treeview></div>`
})
export class AppComponent {

```



```

// Data source for TreeView component
public Countries: Object[] = [
    { id: 1, name: 'India', hasChild: true },
    { id: 2, pid: 1, name: 'Assam' },
    { id: 3, pid: 1, name: 'Bihar' },
    { id: 4, pid: 1, name: 'Tamil Nadu' },
    { id: 6, pid: 1, name: 'Punjab' },
    { id: 7, name: 'Brazil', hasChild: true },
    { id: 8, pid: 7, name: 'Paraná' },
    { id: 9, pid: 7, name: 'Ceará' },
    { id: 10, pid: 7, name: 'Acre' },
    { id: 11, name: 'France', hasChild: true },
    { id: 12, pid: 11, name: 'Pays de la Loire' },
    { id: 13, pid: 11, name: 'Aquitaine' },
    { id: 14, pid: 11, name: 'Brittany' },
    { id: 15, pid: 11, name: 'Lorraine' },
    { id: 16, name: 'Australia', hasChild: true },
    { id: 17, pid: 16, name: 'New South Wales' },
    { id: 18, pid: 16, name: 'Victoria' },
    { id: 19, pid: 16, name: 'South Australia' },
    { id: 20, pid: 16, name: 'Western Australia' },
    { id: 21, name: 'China', hasChild: true },
    { id: 22, pid: 21, name: 'Guangzhou' },
    { id: 23, pid: 21, name: 'Shanghai' },
    { id: 24, pid: 21, name: 'Beijing' },
    { id: 25, pid: 21, name: 'Shantou' }
]

public field:Object={  dataSource: this.Countries, id: 'id', text:
'name', parentID: 'pid', hasChildren: 'hasChild' };
public newData?: any;
public expandedNodes: Set<number> = new Set<number>();
@ViewChild ('treevalidate') tree?: TreeViewComponent;
public onNodeExpand(args: NodeExpandEventArgs | any): void {
    if (args.isInteracted) {
        const nodeId: number = parseInt(args.nodeData['id']);
        if (!this.expandedNodes.has(nodeId)) {
            let childData: any = new
DataManager(this.newData).executeLocal(
                new Query().where(
                    ((this.tree as TreeViewComponent).fields as
FieldsSettingsModel)
                        .parentID as string,
                        'equal',
                        nodeId,
                        false
                    )
                );
            this.tree?.addNodes(childData, args.node, null as any);
            this.expandedNodes.add(nodeId);
        }
    }
}

public onCreate(args: any){
    this.newData = this.tree?.fields.dataSource;
    // Selects the first level nodes alone

```

```

        let resultData = new DataManager(this.newData).executeLocal(new
Query().where(((this.tree as TreeViewComponent ).fields as
FieldsSettingsModel).parentID as string, 'isnull', undefined, false));
        let name = [];
        for (let i = 0; i < resultData.length; i++){
            name.push(((resultData)[i] as Object | any)[(this.tree as
any).fields.text as any]);
        }
        name.sort();
        let arr = [];
        for (let j = 0; j < name.length; j++) {
            let sortedData = new DataManager(this.newData).executeLocal(new
Query().where(((this.tree as TreeViewComponent ).fields as
FieldsSettingsModel).text as string, 'equal', name[j], false));
            let childData = new DataManager(this.newData).executeLocal(new
Query().where(((this.tree as TreeViewComponent ).fields as
FieldsSettingsModel).parentID as string, 'equal', parseInt((sortedData[0] as
Object[] | any)[(this.tree as TreeViewComponent ).fields as
FieldsSettingsModel).id as string]), false));
            arr.push(sortedData[0]);
        }
        // Renders treeview with sorted Nodes
        this.changeDataSource(arr);
        this.tree?.dataBind();
    }
    public changeDataSource(data: Object[]){
        (this.tree as TreeViewComponent).fields = {
            dataSource: data, id: 'id', text: 'name', parentID: 'pid',
            hasChildren: 'hasChild'
        } as FieldsSettingsModel;
    }
}

```

MAIN.TS

```

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

```

Remove parent checkbox in Angular Treeview component

By enabling the `showCheckBox` property, you can render check box before each node of TreeView. However, some application needs to render check box in child nodes alone. In such case, you can remove the check box of the parent node by customizing the CSS.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, Inject, ViewChild } from '@angular/core';
import { TreeViewComponent, DrawNodeEventArgs } from '@syncfusion/ej2-
angular-navigations';
/**

```

```

* Removing checkbox of parent nodes TreeView sample
*/
@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div id='treeparent'><ejs-treeview id='treeElement'
#treevalidate [fields]='field' cssClass="custom" [showCheckBox]=true></ejs-
treeview></div>`
})
export class AppComponent {
  // Data source for TreeView component
  public Countries: Object[] = [
    { id: 1, name: 'India', hasChild: true, expanded: true },
    { id: 2, pid: 1, name: 'Assam' },
    { id: 3, pid: 1, name: 'Bihar' },
    { id: 4, pid: 1, name: 'Tamil Nadu' },
    { id: 6, pid: 1, name: 'Punjab' },
    { id: 7, name: 'Brazil', hasChild: true },
    { id: 8, pid: 7, name: 'Paraná' },
    { id: 9, pid: 7, name: 'Ceará' },
    { id: 10, pid: 7, name: 'Acre' },
    { id: 11, name: 'France', hasChild: true },
    { id: 12, pid: 11, name: 'Pays de la Loire' },
    { id: 13, pid: 11, name: 'Aquitaine' },
    { id: 14, pid: 11, name: 'Brittany' },
    { id: 15, pid: 11, name: 'Lorraine' },
    { id: 16, name: 'Australia', hasChild: true },
    { id: 17, pid: 16, name: 'New South Wales' },
    { id: 18, pid: 16, name: 'Victoria' },
    { id: 19, pid: 16, name: 'South Australia' },
    { id: 20, pid: 16, name: 'Western Australia' },
    { id: 21, name: 'China', hasChild: true },
    { id: 22, pid: 21, name: 'Guangzhou' },
    { id: 23, pid: 21, name: 'Shanghai' },
    { id: 24, pid: 21, name: 'Beijing' },
    { id: 25, pid: 21, name: 'Shantou' }
  ]
  public field:Object = {  dataSource: this.Countries, id: 'id', text:
'name', parentID: 'pid', hasChildren: 'hasChild' };
}

```

MAIN.TS

```

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

```

Get dynamic icon in Angular Treeview component

In TreeView component, you can get the original bound data using the `getTreeData` method. For this method, if you pass the id of the tree node, it returns the corresponding node information, or otherwise

the overall tree nodes information will be returned. You can use this method to get the bound iconCss class in the `nodeChecking` event. Please refer to the following sample.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, Inject, ViewChild } from '@angular/core';
import { NodeCheckEventArgs, TreeViewComponent } from '@syncfusion/ej2-angular-navigations';

/**
 * Icon css sample
 */
@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div id='treeparent'><ejs-treeview id='treeElement'
#treevalidate [fields]='field' (nodeChecking)= 'onNodeCheck($event)'
[showCheckBox]=true [autoCheck]=false></ejs-treeview></div>`
})
export class AppComponent {
  // Data source for TreeView component
  public treeData: Object[] = [
    {
      "nodeId": "01", "nodeText": "Music", "icon": "folder", "expanded":
true, "nodeChild": [
        { "nodeId": "01-01", "nodeText": "Gouttes.mp3", "icon": "audio" }
      ]
    },
    {
      "nodeId": "02", "nodeText": "Videos", "icon": "folder", "expanded":
true, "nodeChild": [
        { "nodeId": "02-01", "nodeText": "Naturals.mp4", "icon": "video"
},
        { "nodeId": "02-02", "nodeText": "Wild.mpeg", "icon": "video" }
      ]
    }
  ];
  public field:Object={  dataSource: this.treeData, id: 'nodeId', text:
'nodeText', child: 'nodeChild', iconCss: 'icon', expanded: 'expanded' };
  @ViewChild('treevalidate') tree?: TreeViewComponent;
  public onNodeCheck(args: NodeCheckEventArgs): void {
    let nodeId: any = args.data[0]['id'];
    // To get the iconCss
    let iconClass = this.tree?.getTreeData(nodeId)[0]['icon'];
    alert('Icon class is ' + iconClass);
  }
}
```

MAIN.TS

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

Hover multi line tree node in Angular Treeview component

This section demonstrates how to hover and select a multi-line tree node. Here, you can set the row height (element class: `e-fullrow`) to be the same as the row content (element class: `e-text-content`)

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, Inject, ViewChild } from '@angular/core';
import { TreeViewComponent, NodeSelectEventArgs } from '@syncfusion/ej2-
angular-navigations';
/**
 * Hovering multiple line treeview
 */
@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div id='treeparent'><ejs-treeview id='treeElement'
#treevalidate [fields]='field' (nodeSelecting)='onSelect($event)'
cssClass="customTree" (created)="onCreate($event)"></ejs-treeview></div>`
})
export class AppComponent {
  // Data source for TreeView component
  public hierarchicalData: Object[] = [
    {
      id: 1, name: 'Web Control sWeb ControlsWeb ControlsWeb
ControlsWeb ControlsWeb ControlsWeb ControlsWeb Controls', expanded: true,
      child: [
        {
          id: 2, name:
'CalendarCalendarCalendarCalendarCalendarCalendarCalendarCalendarCale
ndarCalendarCalendarCalendar', child: [
            { id: 7, name: 'Constructors' },
            { id: 8, name: 'Properties' },
            { id: 9, name: 'Methods' },
            { id: 10, name: 'Events' }
          ]
        },
        {
          id: 3, name: 'Data Grid', child: [
            { id: 11, name: 'Constructors' },
            { id: 12, name: 'Fields' },
            { id: 13, name: 'Properties' },
            { id: 14, name: 'Methods' },
            { id: 15, name: 'Events' }
          ]
        }
      ]
    }
  ]
}
```

```

        },
        {
            id: 4, name: 'DropDownList', child: [
                { id: 16, name: 'Constructors' },
                { id: 17, name: 'Properties' },
                { id: 18, name: 'Methods' }
            ]
        },
        {
            id: 5, name: 'Menu', child: [
                { id: 19, name: 'Constructors' },
                { id: 20, name: 'Fields' },
                { id: 21, name: 'Properties' },
                { id: 22, name: 'Methods' },
                { id: 23, name: 'Events' }
            ]
        }
    ],
    {
        id: 24, name: 'Web Controls',
        child: [
            {
                id: 25, name: 'Calendar', child: [
                    { id: 26, name: 'Constructors' },
                    { id: 27, name: 'Properties' },
                    { id: 28, name: 'Methods' },
                    { id: 29, name: 'Events' }
                ]
            },
            {
                id: 30, name: 'Data Grid', child: [
                    { id: 31, name: 'Constructors' },
                    { id: 32, name: 'Fields' },
                    { id: 33, name: 'Properties' },
                    { id: 34, name: 'Methods' },
                    { id: 35, name: 'Events' }
                ]
            }
        ]
    }
];
public field:Object = { dataSource: this.hierarchicalData, id: 'id',
text: 'name', child: 'child' };
@ViewChild('treevalidate') tree?: TreeViewComponent;
// Triggers on node selection
public onSelect(args: NodeSelectEventArgs): void {
    this.setHeight(args.node);
}
public onCreate(args: any) {
    // Triggers on mouse hover/keydown event
    ['mouseover', 'keydown'].forEach( evt =>
        this.tree?.element.addEventListener(evt,
            (event) => { this.setHeight(event.target); }));
}
// Sets e-fullrow to be the same as e-text-content
public setHeight(element: any) {

```

```

        if(this.tree?.fullRowSelect) {
            if(element?.classList.contains("e-treeview")) {
                element = element.querySelector(".e-node-focus").querySelector(".e-
fullrow");
            }
            else if(element.classList.contains("e-list-parent")) {
                element = element.querySelector(".e-fullrow");
            }
            else if(element.classList.value != ("e-fullrow") &&
element.closest(".e-list-item")) {
                element = element.closest(".e-list-item").querySelector(".e-
fullrow");
            }
            if(element.nextElementSibling)
                element.style.height = element.nextElementSibling.offsetHeight
+"px";
        }
    }
}

```

MAIN.TS

```

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

```

Select one child in Angular Treeview component

TreeView allows both single and multiple selections. If your application needs to select one child at a time under one specific parent, refer to the following example. Here, you can achieve this in the `nodeSelecting` event of TreeView. However, you can reset the selected child and make another selection by pressing Ctrl + selected nodes.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, Inject, ViewChild } from '@angular/core';
import { TreeViewComponent, NodeSelectEventArgs } from '@syncfusion/ej2-
angular-navigations';
/**
 * Single child selection at a time
 */
@Component({
  imports: [
    FormsModule,TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div id='treeparent'><ejs-treeview #tree id="listtree"
allowMultiSelection='allowMultiSelection' [fields]='listfields'
(nodeSelecting)="onNodeSelecting($event)"></ejs-treeview></div>`
})

```

```

export class AppComponent {
  @ViewChild('tree')
  public tree?: TreeViewComponent;
  // Self-referential list data source for TreeView component
  public localData: Object[] = [
    { id: 1, name: 'Parent 1', hasChild: true, expanded: true },
    { id: 2, pid: 1, name: 'Child 1' },
    { id: 3, pid: 1, name: 'Child 2' },
    { id: 4, pid: 1, name: 'Child 3' },
    { id: 7, name: 'Parent 2', hasChild: true, expanded: true },
    { id: 8, pid: 7, name: 'Child 1' },
    { id: 9, pid: 7, name: 'Child 2' },
    { id: 10, pid: 7, name: 'Child 3' },
  ];
  public listfields: Object = { dataSource: this.localData, id: 'id',
parentID: 'pid', text: 'name', hasChildren: 'hasChild' };
  public allowMultiSelection: boolean = true;
  public parent?: any; public child?: any;
  public count: boolean = false;
  public childCount: boolean = false;
  // Triggers when you select any node
  public onNodeSelecting(args: NodeSelectEventArgs): void {
    console.log(args.nodeData);
    let id: any = args.nodeData['parentID'];
    if (!this.count) {
      this.parent = id;
      this.count = true;
    }
    if (!this.childCount) {
      this.child = args.nodeData['id'];
      this.childCount = true
    }
    if (id !== null && id === this.parent) {
      let element: HTMLElement = this.tree?.element.querySelector('[data-
uid="' + id + '"]') as HTMLElement;
      let liElements: any = element.querySelectorAll('ul li');
      for (let i: number = 0; i < liElements.length; i++) {
        let nodeData: any = this.tree?.getNode(liElements[i]);
        if (nodeData.selected && args.action === "select" && this.child !==
args.nodeData['id']) {
          args.cancel = true;
        }
        // For unselect the selectedNodes
        else if (args.action === "un-select" && this.child ===
args.nodeData['id']) {
          this.childCount = false;
          this.child = null;
          this.parent = null;
          this.count = false;
        }
      }
    } else if (id !== this.parent && id !== null) {
      if(args.action === "select"){
        args.cancel = true
      }
    } else if (id === null){
      this.childCount = false;
    }
  }
}

```



```

        this.child = null;
        this.parent = null;
        this.count = false
    }
}
}

```

MAIN.TS

```

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

```

Get all child nodes in Angular Treeview component

This section demonstrates how to get the child nodes from corresponding parent ID. Using the `getNode` method, you can get the node details of TreeView. Please refer to the following sample.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, Inject, ViewChild } from '@angular/core';
import { TreeViewComponent } from '@syncfusion/ej2-angular-navigations';
/**
 * Treeview sample for getting child details via parent ID
 */
@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div id='treeparent'><ejs-treeview id='treeElement'
#treevalidate [fields]='field' (created)='onCreate($event)'
[loadOnDemand]=false></ejs-treeview></div><input type="text" class="e-input"
id="Nodes" style="margin-left: 10px;margin-top:10px; width: 175px;"
placeholder="Enter the parent ID( Ex: AS)" />
    <input type="button" class="btn btn-primary" value="Submit" id="btn"
/>`
})
export class AppComponent {
  // Data source for TreeView component
  public data: Object[] = [
    {
      code: "AF", name: "Africa", countries: [
        { code: "NGA", name: "Nigeria" },
        { code: "EGY", name: "Egypt" },
        { code: "ZAF", name: "South Africa" }
      ]
    },
    {
      code: "AS", name: "Asia", countries: [

```

```

        { code: "CHN", name: "China" },
        { code: "IND", name: "India", countries: [
            { code: "TN", name: "TamilNadu" }
        ]},
        { code: "JPN", name: "Japan" }
    ]
},
{
    code: "EU", name: "Europe", countries: [
        { code: "DNK", name: "Denmark" },
        { code: "FIN", name: "Finland" },
        { code: "AUT", name: "Austria" }
    ]
},
{
    code: "NA", name: "North America", countries: [
        { code: "USA", name: "United States of America" },
        { code: "CUB", name: "Cuba" },
        { code: "MEX", name: "Mexico" }
    ]
},
{
    code: "SA", name: "South America", countries: [
        { code: "BR", name: "Brazil" },
        { code: "COL", name: "Colombia" },
        { code: "ARG", name: "Argentina" }
    ]
},
];
public field:Object = {  dataSource: this.data, id: 'code', text: 'name',
child: 'countries' };
@ViewChild ('treevalidate') tree?: TreeViewComponent;
public onCreate(args: any): void {
    let proxy = this.tree;
    (document.getElementById("btn") as
HTMLElement).addEventListener("click", (event)=>{
        let id = (document.getElementById('Nodes') as HTMLElement |
any).value
        let element= proxy?.element.querySelector('[data-uid="' + id +
'" ]');
        // Gets the child Element
        let liElements = element?.querySelectorAll('ul li');
        let arr= [];
        for (let i = 0; i < (liElements as NodeListOf<Element>).length;
i++) {
            let nodeData= proxy?.getNode((liElements as
NodeListOf<Element>)[i]);
            arr.push(nodeData);
        }
        alert(JSON.stringify(arr));
    });
}
}

```

MAIN.TS

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

Disable checkbox of the tree node in Angular Treeview component

You can disable the check box alone in TreeView instead of disabling the whole node. You need to include the `e-checkbox-disabled` class into the check box element using the `drawNode` event. Please refer to the following sample to disable the check box of the tree nodes.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TreeViewModule } from '@syncfusion/ej2-angular-navigations'
import { Component, Inject, ViewChild } from '@angular/core';
import { TreeViewComponent, DrawNodeEventArgs } from '@syncfusion/ej2-
angular-navigations';
/**
 * Treeview Disable check box of parent nodes sample
 */
@Component({
  imports: [
    FormsModule, TreeViewModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div id='treeparent'><ejs-treeview id='treeElement'
#treevalidate [fields]='field' (drawNode)='drawNode($event)'
[showCheckBox]=true></ejs-treeview></div>`
})
export class AppComponent {
  // Data source for TreeView component
  public Countries: Object[] = [
    { id: 1, name: 'India', hasChild: true, expanded: true },
    { id: 2, pid: 1, name: 'Assam' },
    { id: 3, pid: 1, name: 'Bihar' },
    { id: 4, pid: 1, name: 'Tamil Nadu' },
    { id: 6, pid: 1, name: 'Punjab' },
    { id: 7, name: 'Brazil', hasChild: true },
    { id: 8, pid: 7, name: 'Paraná' },
    { id: 9, pid: 7, name: 'Ceará' },
    { id: 10, pid: 7, name: 'Acre' },
    { id: 11, name: 'France', hasChild: true },
    { id: 12, pid: 11, name: 'Pays de la Loire' },
    { id: 13, pid: 11, name: 'Aquitaine' },
    { id: 14, pid: 11, name: 'Brittany' },
    { id: 15, pid: 11, name: 'Lorraine' },
    { id: 16, name: 'Australia', hasChild: true },
    { id: 17, pid: 16, name: 'New South Wales' },
    { id: 18, pid: 16, name: 'Victoria' },
    { id: 19, pid: 16, name: 'South Australia' },
    { id: 20, pid: 16, name: 'Western Australia' },
    { id: 21, name: 'China', hasChild: true },
    { id: 22, pid: 21, name: 'Guangzhou' },
```

```

    { id: 23, pid: 21, name: 'Shanghai' },
    { id: 24, pid: 21, name: 'Beijing' },
    { id: 25, pid: 21, name: 'Shantou' }
  ]
  public field:Object = { dataSource: this.Countries, id: 'id', text:
'name', parentID: 'pid', hasChildren: 'hasChild' };
  @ViewChild('treevalidate') tree?: TreeViewComponent;
  // Disables the checkbox alone in treeview
  public drawNode(args: DrawNodeEventArgs): void {
    let ele: HTMLElement = args.node.querySelector('.e-checkbox-wrapper')
  as HTMLElement;
    ele.classList.add('e-checkbox-disabled');
  }
}

```

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 Treeview component

This article describes the API migration process of TreeView component from Essential JS 1 to Essential JS 2.

Add nodes

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Add node | **Method:** `addNode`
`<ej-treeview id='tree' [fields]='field'></ej-treeview>
</><Script>var treeObj = $("#tree").data("ejTreeView");
treeObj.addNode("Node", "#book");` | **Method:** `addNodes`
`<ejs-treeview id='tree' [fields]='field'></ejs-treeview>
</><Script>@ViewChild("tree") tree: TreeViewComponent;
var object = [{ id: "temp", name: "New node" }, { id: "new", name: "New node 1" }];
tree.addNodes(object, "book");` |

| Triggers before adding node | **Event:** `beforeAdd`
`(beforeAdd)="beforeAdd()" [fields]='field'></ej-treeview>
</><Script>beforeAdd() {}` |
 Not Applicable |

| Adding node after a particular node | **Method:** `insertAfter`
`<ej-treeview id='tree' [fields]='field'></ej-treeview>
</><Script>var treeObj = $("#tree").data("ejTreeView");
treeObj.insertAfter("node", "book");` | **Can be achieved using**
`<ejs-treeview id='tree' [fields]='field'></ejs-treeview>
</><Script>@ViewChild("tree") tree: TreeViewComponent;
var object = [{ id: "1", name: "node" }];
var child = tree.element.querySelector("[data-uid="book"]);
var parent = child.parentElement.closest('.e-list-item');
var level = parseInt(parent.getAttribute('aria-level'))+1;
var childNodes = Array.from(parent.querySelectorAll('.e-list-item.e-level-`

```
'+level))<br/>var index = childNodes.indexOf(child)<br/>tree.addNodes(object, "book",
index+1); |
```

| Adding node before a particular node | **Method:** *insertBefore*

<ej-treeview id='tree'
[fields]='field'></ej-treeview>

Script
var treeObj =
\$("#tree").data("ejTreeView");
treeObj.insertBefore("node", "book"); | **Can be achieved**
using

<ejs-treeview id='tree' [fields]='field'></ejs-treeview>

Script
@ViewChild("tree") tree: TreeViewComponent;
var object = [{ id: "1", name: "node"
}],
var child = tree.element.querySelector("[data-uid="book"]);
var parent =
child.parentElement.closest('.e-list-item');
var level = parseInt(parent.getAttribute('aria-
level'))+1;
var childNodes = Array.from(parent.querySelectorAll('.e-list-item.e-level-
' + level))
var index = childNodes.indexOf(child)
tree.addNodes(object, "book", index-1);
|

| Triggers when node is added successfully | **Event:** *nodeAdd*

<ej-treeview id='tree'
(nodeAdd)="nodeAdd()" [fields]='field'></ej-treeview>

Script
nodeAdd() {} |
Event: *dataSourceChanged*

<ejs-treeview id='tree'
(dataSourceChanged)="dataSourceChanged()" [fields]='field'></ejs-treeview>

>**Script**
public dataSourceChanged(): void {} |

Common

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Keyboard Navigation | **Property:** *allowKeyboardNavigation*

<ej-treeview id='tree'
allowKeyboardNavigation=false [fields]='field'></ej-treeview> | **Can be achieved**
using,

<ejs-treeview id='tree' (keyPress)="keyPress(\$event)"></ejs-treeview>

>**Script**
public keyPress(args): void {
args.cancel = true;
} |

| Triggers before node is cut | **Event:** *beforeCut*

<ej-treeview id='tree'
(beforeCut)="beforeCut()" [fields]='field'></ej-treeview>

Script
beforeCut() {} |
Not Applicable |

| Triggers before node is deleted | **Event:** *beforeDelete*

<ej-treeview id='tree'
(beforeDelete)="beforeDelete()" [fields]='field'></ej-treeview>

Script
beforeDelete() {} | Not Applicable |

| Triggers before loading nodes | **Event:** *beforeLoad*

<ej-treeview id='tree'
(beforeLoad)="beforeLoad()" [fields]='field'></ej-treeview>

Script
beforeLoad()
{ } | Not Applicable |

| Triggers before node is pasted | **Event:** *beforePaste*

<ej-treeview id='tree'
(beforePaste)="beforePaste()" [fields]='field'></ej-treeview>

Script
beforePaste() {} | Not Applicable |

| Triggers when Treeview is created | **Event:** *create*

<ej-treeview id='tree'
(create)="onCreate()" [fields]='field'></ej-treeview>

Script
onCreate() {} | **Event:**
created

<ejs-treeview id='tree' (created)="onCreated()"></ejs-treeview>

>**Script**
public onCreated(): void {} |

| Css class | **Property:** `cssClass`
`<ej-treeview id='tree' cssClass='custom' [fields]='field'></ej-treeview>` | **Property:** `cssClass`
`<ej-treeview id='tree' cssClass='custom'></ej-treeview>` |

| Triggers when Treeview is destroyed | **Event:** `destroy`
`(destroy)="onDestroy()" [fields]='field'></ej-treeview>
</>Script
onDestroy() {}` | **Event:** `destroyed`
`(destroyed)="onDestroyed()"></ej-treeview>
</>Script
public onDestroy(): void {}` |

| Destroy Treeview control | **Method:** `destroy`
`[fields]='field'></ej-treeview>
</>Script
var treeObj = $("#tree").data("ejTreeView");
treeObj.destroy();` | **Method:** `destroy`
`[fields]='field'></ej-treeview>
</>Script
@ViewChild("tree") tree: TreeViewComponent;
tree.destroy();` |

| Disable Node | **Method:** `disableNode`
`[fields]='field'></ej-treeview>
</>Script
var treeObj = $("#tree").data("ejTreeView");
treeObj.disableNode("1");` | **Method:** `disableNodes`
`[fields]='field'></ej-treeview>
</>Script
@ViewChild("tree") tree: TreeViewComponent;
tree.disableNodes(["1", "2"]);` |

| Enable Animation | **Property:** `enableAnimation`
`enableAnimation=false [fields]='field'></ej-treeview>` | **Property:** `animation`
`<ej-treeview id='tree' [fields]='field'></ej-treeview>
</>Script
var treeObj = document.getElementById('tree').ej2_instances[0]
treeObj.animation.expand.duration=0
treeObj.animation.collapse.duration=0` |

| Control state | **Property:** `enabled`
`enabled=false [fields]='field'></ej-treeview>` | Not Applicable |

| Enable Node | **Method:** `enableNode`
`[fields]='field'></ej-treeview>
</>Script
var treeObj = $("#tree").data("ejTreeView");
treeObj.enableNode("1");` | **Method:** `enableNodes`
`[fields]='field'></ej-treeview>
</>Script
@ViewChild("tree") tree: TreeViewComponent;
tree.enableNodes(["1", "2"]);` |

| Persistence | **Property:** `enablePersistence`
`enablePersistence=true [fields]='field'></ej-treeview>` | **Property:** `enablePersistence`
`<ej-treeview id='tree' enablePersistence=true [fields]='field'></ej-treeview>` |

| Right to Left | **Property:** `enableRTL`
`enableRTL=true [fields]='field'></ej-treeview>` | **Property:** `enableRtl`
`<ej-treeview id='tree' enableRtl=true [fields]='field'></ej-treeview>` |

| Ensure visibility | **Method:** `ensureVisible`
`[fields]='field'></ej-treeview>
</>Script
var treeObj = $("#tree").data("ejTreeView");
treeObj.ensureVisible("1");` | **Method:** `ensureVisible`
`[fields]='field'></ej-treeview>
</>Script
@ViewChild("tree") tree: TreeViewComponent;
tree.ensureVisible("1");` |

| Mapping fields | **Property:** `fields`
`<ej-treeview id='tree' [fields]='field'></ej-treeview>
</>Script
 public fields:any = {
 id: "id",text: "text",parentId: "parent",dataSource: this.localData,isChecked: "checked",selected: "selected",spriteCssClass: "spriteImage",imageUrl: "imageUrl",htmlAttribute: "nodeProperty",linkAttribute: "linkProperty",imageAttribute: "imageProperty"></> | Property: fields
<ej-treeview id='tree' [fields]='field'></ej-treeview>
</>Script
 public field: Object = {
 dataSource: this.localData, id: 'id', parentId: 'pid', text: 'name', hasChildren: 'hasChild', expanded: 'expanded', htmlAttribute: 'htmlAttributes', imageAttribute: 'img', imageUrl: 'imageUrl', isChecked: 'checked', linkAttribute: 'linkAttribute', query: null, selected: 'selected', spriteCssClass: 'custom', tableName: null }; |`

| Get child nodes | **Method:** `getChildren`
`<ej-treeview id='tree' [fields]='field'></ej-treeview>
</>Script
 var treeObj = $("#tree").data("ejTreeView");
 treeObj.getChildren("1"); | Can be achieved using,
<ej-treeview id='tree' [fields]='field'></ej-treeview>
</>Script
 @ViewChild("tree") tree: TreeViewComponent;
 var parent=tree.element.querySelector('[data-uid="1"]')
 console.log(parent.querySelector('.e-list-item')) |`

| Get node | **Method:** `getNode`
`<ej-treeview id='tree' [fields]='field'></ej-treeview>
</>Script
 var treeObj = $("#tree").data("ejTreeView");
 treeObj.getNode("1"); | Method: getNode
<ej-treeview id='tree' [fields]='field'></ej-treeview>
</>Script
 @ViewChild("tree") tree: TreeViewComponent;
 tree.getNode("1"); |`

| Get node by index | **Method:** `getNodeByIndex`
`<ej-treeview id='tree' [fields]='field'></ej-treeview>
</>Script
 var treeObj = $("#tree").data("ejTreeView");
 treeObj.getNodeByIndex(3); | Can be achieved using,
<ej-treeview id='tree' [fields]='field'></ej-treeview>
</>Script
 @ViewChild("tree") tree: TreeViewComponent;
 var nodes=tree.element.querySelectorAll('.e-list-item')
 console.log(nodes[3]); |`

| Get node count | **Method:** `getNodeCount`
`<ej-treeview id='tree' [fields]='field'></ej-treeview>
</>Script
 var treeObj = $("#tree").data("ejTreeView");
 treeObj.getNodeCount(); | Can be achieved using,
<ej-treeview id='tree' [fields]='field'></ej-treeview>
</>Script
 @ViewChild("tree") tree: TreeViewComponent;
 var nodes=tree.element.querySelectorAll('.e-list-item')
 console.log("Node count is " + nodes.length); |`

| Get node index | **Method:** `getNodeIndex`
`<ej-treeview id='tree' [fields]='field'></ej-treeview>
</>Script
 var treeObj = $("#tree").data("ejTreeView");
 treeObj.getNodeIndex("book"); | Can be achieved using,
<ej-treeview id='tree' [fields]='field'></ej-treeview>
</>Script
 @ViewChild("tree") tree: TreeViewComponent;
 var nodes=tree.element.querySelectorAll('.e-list-item')
 var node = tree.element.querySelector('[data-uid="'+ "book" + "']')
 while(i<nodes.length)
 {
 if(nodes[i] === node) {console.log("Node index is "+i)
 break;
 }
 i++
 } |`

| Get parent of node | **Method:** `getParent`
`<ej-treeview id='tree' [fields]='field'></ej-treeview>
</>Script
 var treeObj = $("#tree").data("ejTreeView");
 treeObj.getParent("book"); |`

```

/>treeObj.getParent("book"); | Can be achieved using,<br><br><ej-treeview id='tree'
[fields]='field'></ej-treeview><br /><br /><Script<br />@ViewChild("tree") tree:
TreeViewComponent;<br>var child=tree.element.querySelector('[data-uid="" + "book" +
""]');<br>var parent = child.parentNode.closest('.e-list-item')<br>console.log(parent) |

| Get tree node text | Method: getText<br><br><ej-treeview id='tree' [fields]='field'></ej-
treeview><br /><br><Script<br>var treeObj = $("#tree").data("ejTreeView");<br
/>treeObj.getText("1"); | Can be achieved using,<br><br><ej-treeview id='tree'
[fields]='field'></ej-treeview><br /><br /><Script<br />@ViewChild("tree") tree:
TreeViewComponent;<br>var nodeText =tree.getNode("1")['text']<br>console.log(nodeText) |

| Get updated datasource | Method: getTreeData<br><br><ej-treeview id='tree'
[fields]='field'></ej-treeview><br /><br><Script<br>var treeObj =
$("#tree").data("ejTreeView");<br />treeObj.getTreeData(); | Method:
getTreeData<br><br><ej-treeview id='tree' [fields]='field'></ej-treeview><br /><br /><Script<br
/>@ViewChild("tree") tree: TreeViewComponent;<br>tree.getTreeData(); |

| Get visible nodes | Method: getVisibleNodes<br><br><ej-treeview id='tree' [fields]='field'></ej-
treeview><br /><br><Script<br>var treeObj = $("#tree").data("ejTreeView");<br
/>treeObj.getVisibleNodes(); | Not Applicable |

| Height of Treeview control | Property: height<br><br><ej-treeview id='tree' height="400px"
[fields]='field'></ej-treeview> | Can be achieved using, <br><br><ej-treeview id='tree'
cssClass='custom'></ej-treeview><br><Css<br>.e-treeview.custom{<br>height: 400px;<br>} |

| Checking for child nodes | Method: hasChildNode<br><br><ej-treeview id='tree'
[fields]='field'></ej-treeview><br /><br><Script<br>var treeObj =
$("#tree").data("ejTreeView");<br />treeObj.hasChildNode("book"); | Can be achieved
using,<br><br><ej-treeview id='tree' [fields]='field'></ej-treeview><br /><br /><Script<br
/>@ViewChild("tree") tree: TreeViewComponent;<br>var
parent=tree.element.querySelector('[data-uid="book"]')<br>if (parent.querySelector('.e-list-
item') !== null) {<br>console.log("Has child node")<br>} |

| Hide all nodes | Method: hide<br><br><ej-treeview id='tree' [fields]='field'></ej-treeview><br
/><br><Script<br>var treeObj = $("#tree").data("ejTreeView");<br />treeObj.hide(); | Can be
achieved using,<br><br><ej-treeview id='tree' [fields]='field'></ej-treeview><br /><br
/><Script<br />@ViewChild("tree") tree:
TreeViewComponent;<br>tree.element.querySelector('.e-list-parent').style.display="none" |

| Hide node | Method: hideNode<br><br><ej-treeview id='tree' [fields]='field'></ej-treeview><br
/><br><Script<br>var treeObj = $("#tree").data("ejTreeView");<br />treeObj.hideNode("book");
| Can be achieved using,<br><br><ej-treeview id='tree' [fields]='field'></ej-treeview><br /><br
/><Script<br />@ViewChild("tree") tree:
TreeViewComponent;<br>tree.element.querySelector('[data-
uid="book"]').style.display="none" |

| HTML Attributes | Property: htmlAttributes<br><br><ej-treeview id='tree'
[fields]='field'[htmlAttributes]="htmlAttr"></ej-treeview><br /><br /><script<br />constructor()

```


{
this.htmlAttr = {name: "treeview"};
} | **Can be achieved using**,

<ejs-treeview id='tree' [fields]='field'></ejs-treeview>

<Script
@ViewChild("tree") tree: TreeViewComponent;
tree.element.classList.add("htmlAttr") |

| To check if child nodes are loaded | **Method:** *isChildLoaded*

<ej-treeview id='tree' [fields]='field'></ej-treeview>

<Script
var treeObj = \$("#tree").data("ejTreeView");
treeObj.isChildLoaded("book"); | **Can be achieved using**,

<ejs-treeview id='tree' [fields]='field'></ejs-treeview>

<Script
@ViewChild("tree") tree: TreeViewComponent;
var parent=tree.element.querySelector("[data-uid="book"]")
if (parent.querySelector('.e-list-item') !== null) {
console.log("Child is loaded")
} |

| To check if node is disabled | **Method:** *isDisabled*

<ej-treeview id='tree' [fields]='field'></ej-treeview>

<Script
var treeObj = \$("#tree").data("ejTreeView");
treeObj.isDisabled("book"); | **Can be achieved using**,

<ejs-treeview id='tree' [fields]='field'></ejs-treeview>

<Script
@ViewChild("tree") tree: TreeViewComponent;
var node=tree.element.querySelector("[data-uid="book"]")
if (node.classList.contains('e-disable') === true) {
console.log("Node is disabled")
} |

| To check if node exists in Treeview | **Method:** *isExist*

<ej-treeview id='tree' [fields]='field'></ej-treeview>

<Script
var treeObj = \$("#tree").data("ejTreeView");
treeObj.isExist("book"); | **Can be achieved using**,

<ejs-treeview id='tree' [fields]='field'></ejs-treeview>

<Script
@ViewChild("tree") tree: TreeViewComponent;
if (tree.getNode('book')['text']) !== "") {
console.log("Node exists")
} |

| To check if node is visible | **Method:** *isVisible*

<ej-treeview id='tree' [fields]='field'></ej-treeview>

<Script
var treeObj = \$("#tree").data("ejTreeView");
treeObj.isVisible("book"); | **Can be achieved using**,

<ejs-treeview id='tree' [fields]='field'></ejs-treeview>

<Script
@ViewChild("tree") tree: TreeViewComponent;
if (tree.element.querySelector("[data-uid="book"]").style.display !== "none"){
console.log("Node is visible")
} |

| Triggers on key press | **Event:** *keyPress*

<ej-treeview id='tree' (keyPress)="onKeyPress()" [fields]='field'></ej-treeview>

<Script
onKeyPress() {} | **Event:** *keyPress*

<ejs-treeview id='tree' (keyPress)="onKeyPress()"></ejs-treeview>

<Script
public onKeyPress(): void {} |

| Load Treeview nodes from particular URI | **Method:** *loadData*

<ej-treeview id='tree' [fields]='field'></ej-treeview>

<Script
var treeObj = \$("#tree").data("ejTreeView");
treeObj.loadData("childData", "book"); | **Can be achieved using**,

<ejs-treeview id='tree' [fields]='field'></ejs-treeview>

<Script
@ViewChild("tree") tree: TreeViewComponent;
var dataManager = new DataManager {
Url = "/FileContent/rootNode",
Adaptor = "UrlAdaptor",
CrossDomain = true};
dataManager.executeQuery(new ej.data.Query().take(8).then((e) => { var childData = e.result;
tree.addNodes(childData, "book")
})); |

| Triggers when data load fails | **Event:** *loadError*
 (loadError)="loadError()" [fields]='field'></ej-treeview>

</Script>
loadError() {} |
Can be achieved using,

<ejs-treeview id='tree' [fields]='field'></ejs-treeview>

</Script>
@ViewChild("tree") tree: TreeViewComponent;
var dataManager = new
 DataManager {
Url = "/FileContent/rootNode",
Adaptor = "UrlAdaptor",

 />CrossDomain = true};
dataManager.executeQuery(new ej.data.Query().take(8).error((e)
 => { console.log('Data load failed')
}); |

| Load on demand | **Property:** *loadOnDemand*
 loadOnDemand=true [fields]='field'></ej-treeview> | **Property:** *loadOnDemand*
 loadOnDemand=false></ej-treeview>
Treeview is rendered in load on demand by default |

| Triggers when data load is success | **Event:** *loadSuccess*
 (loadSuccess)="loadSuccess()" [fields]='field'></ej-treeview>

</Script>

 loadSuccess() {}
@ViewChild("tree") tree:
 TreeViewComponent;
tree.loadData("childData", \$("#book")); | **Can be achieved**
using,

<ejs-treeview id='tree' [fields]='field'></ejs-treeview>

</Script>

 @ViewChild("tree") tree: TreeViewComponent;
var dataManager = new DataManager
 {
Url = "/FileContent/rootNode",
Adaptor = "UrlAdaptor",
CrossDomain =
 true};
dataManager.executeQuery(new ej.data.Query().take(8).then((e) => {
 console.log('Data loaded successfully')
}); |

| To move node | **Method:** *moveNode*
 <ej-treeview id='tree' [fields]='field'></ej-treeview>

</Script>
var treeObj = \$("#tree").data("ejTreeView");

 />treeObj.moveNode("book", "art"); | **Method:** *moveNodes*
 <ejs-treeview id='tree' [fields]='field'></ej-treeview>

</Script>
@ViewChild("tree") tree:
 TreeViewComponent;
tree.moveNodes(["book"], "art"); |

| Triggers when node is clicked successfully | **Event:** *nodeClick*
 (nodeClick)="nodeClick()" [fields]='field'></ej-treeview>

</Script>
nodeClick() {} |
Event: *nodeClicked*
 (nodeClicked)="nodeClicked()"></ej-treeview>

</Script>
public nodeClicked(): void {} |

| Triggers when node is cut successfully | **Event:** *nodeCut*
 (nodeCut)="nodeCut()" [fields]='field'></ej-treeview>

</Script>
nodeCut() {} | Not
 Applicable |

| Triggers when node is deleted successfully | **Event:** *nodeDelete*
 (nodeDelete)="nodeDelete()" [fields]='field'></ej-treeview>

</Script>
nodeDelete()
 {} | **Event:** *dataSourceChanged*
 (dataSourceChanged)="dataSourceChanged()" [fields]='field'></ej-treeview>

 /></Script>
public dataSourceChanged(): void {} |

| Triggers when node is pasted successfully | **Event:** *nodePaste*
 (nodePaste)="nodePaste()" [fields]='field'></ej-treeview>

</Script>
nodePaste() {} |
 Not Applicable |

| Triggers when nodes are loaded successfully | **Event:** *ready*
 (ready)="onReady()" [fields]='field'></ej-treeview>

<Script>
onReady() {} | **Event:** *dataBound*

<ejs-treeview id='tree' (dataBound)="dataBound()"></ejs-treeview>

<Script>
public dataBound(): void {} |

| Refresh Treeview control | **Method:** *refresh*
 <ej-treeview id='tree' [fields]='field'></ej-treeview>

<Script>
var treeObj = \$("#tree").data("ejTreeView");
treeObj.refresh(); | Not Applicable |

| To show all nodes | **Method:** *show*
 <ej-treeview id='tree' [fields]='field'></ej-treeview>

<Script>
var treeObj = \$("#tree").data("ejTreeView");
treeObj.show(); | **Can be achieved using,**
 <ejs-treeview id='tree' [fields]='field'></ejs-treeview>

<Script>
@ViewChild("tree") tree: TreeViewComponent;
tree.element.querySelector('.e-list-parent').style.display="block" |

| Show node | **Method:** *showNode*
 <ej-treeview id='tree' [fields]='field'></ej-treeview>

<Script>
var treeObj = \$("#tree").data("ejTreeView");
treeObj.showNode("book"); | **Can be achieved using,**
 <ejs-treeview id='tree' [fields]='field'></ejs-treeview>

<Script>
@ViewChild("tree") tree: TreeViewComponent;
tree.element.querySelector('[data-uid="book"]').style.display="block" |

| Remove all Treeview nodes | **Method:** *removeAll*
 <ej-treeview id='tree' [fields]='field'></ej-treeview>

<Script>
var treeObj = \$("#tree").data("ejTreeView");
treeObj.removeAll(); | **Can be achieved using,**
 <ejs-treeview id='tree' [fields]='field'></ejs-treeview>

<Script>
@ViewChild("tree") tree: TreeViewComponent;
tree.removeNodes([tree.element.querySelector('.e-list-parent')]) |

| Remove Treeview node | **Method:** *removeNode*
 <ej-treeview id='tree' [fields]='field'></ej-treeview>

<Script>
var treeObj = \$("#tree").data("ejTreeView");
treeObj.removeNode("book"); | **Method:** *removeNodes*
 <ejs-treeview id='tree' [fields]='field'></ejs-treeview>

<Script>
@ViewChild("tree") tree: TreeViewComponent;
tree.removeNodes("book"); |

| Sort order | **Property:** *sortSettings*
 <ej-treeview id='tree' [fields]='field'></ej-treeview>

<Script>
\$("#tree").ejTreeView({
fields: {dataSource: treeData, id: "id", parentId: "pid", text: "name", hasChild: "hasChild"},
sortSettings: { allowSorting: true, sortOrder: ej.sortOrder.Descending }
}); | **Property:** *sortOrder*
 <ejs-treeview id='tree' [fields]='field' sortOrder='Descending'></ejs-treeview> |

| Update node text | **Method:** *updateText*
 <ej-treeview id='tree' [fields]='field'></ej-treeview>

<Script>
var treeObj = \$("#tree").data("ejTreeView");
treeObj.updateText("book", "text"); | **Method:** *updateNode*
 <ejs-treeview id='tree' [fields]='field'></ejs-treeview>

<Script>
@ViewChild("tree") tree: TreeViewComponent;
tree.updateNode("book", "text"); |

| Width of Treeview control | **Property:** *width*
 <ej-treeview id='tree' width="300px" [fields]='field'></ej-treeview> | **Can be achieved using,**
 <ejs-treeview id='tree' cssClass='custom'></ejs-treeview>
<Css>
.e-treeview.custom{
width: 300px;
} |

CheckBox

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Prevent auto-check of child and parent | **Property:** `autoCheck`
`<ej-treeview id='tree' showCheckbox=true autoCheck=false [fields]='field'></ej-treeview>` | **Property:** `autoCheck`
`<ej-treeview id='tree' autoCheck=false showCheckBox=true [fields]='field'></ej-treeview>` |

| Prevent indeterminate state in parent node | **Property:** `autoCheckParentNode`
`<ej-treeview id='tree' autoCheckParentNode=true showCheckbox=true [fields]='field'></ej-treeview>` | **Can be achieved using,**
`<ejs-treeview id='tree' nodeChecked="nodeChecked" [fields]='field' autoCheck=false showCheckBox=true></ejs-treeview>`
Script

```
@ViewChild("tree") tree: TreeViewComponent;
public nodeChecked(args): void {
  var child = tree.element.querySelector("[data-uid='" + args.data[0]['id'] + "']");
  var checkNodes = [];
  var element = child.parentNode;
  while ((element !== null) || (element !== undefined) && !element.parentNode.classList.contains('e-treeview')) {
    element = element.parentNode;
    var id = element.getAttribute('data-uid');
    if (id !== null) checkNodes.push(element.getAttribute('data-uid'));
    if (child.querySelector('.e-list-item') !== null && args.isInteracted === true && args.action === 'check') {
      tree.autoCheck = true;
      tree.checkAll(child.getAttribute('data-uid'));
    } else if (child.querySelector('.e-list-item') !== null && args.isInteracted === true && args.action === 'uncheck') {
      tree.autoCheck = true;
      tree.uncheckAll(child.getAttribute('data-uid'));
    }
    tree.autoCheck = false;
    if (args.action === 'check') {
      tree.checkAll(checkNodes);
    } else if (args.action === 'uncheck' && child.parentNode.querySelector('.e-check') === null) {
      tree.uncheckAll(checkNodes);
    }
  }
}
```

 |

| Check all nodes | **Method:** `checkAll`
`<ej-treeview id='tree' [fields]='field' showCheckbox=true></ej-treeview>`
Script

```
var treeObj = $("#tree").data("ejTreeView");
treeObj.checkAll();
```

 | **Method:** `checkAll`
`<ejs-treeview id='tree' showCheckBox=true [fields]='field'></ejs-treeview>`
Script

```
@ViewChild("tree") tree: TreeViewComponent;
tree.checkAll();
```

 |

| Check node | **Method:** `checkNode`
`<ej-treeview id='tree' [fields]='field' showCheckbox=true></ej-treeview>`
Script

```
var treeObj = $("#tree").data("ejTreeView");
treeObj.checkNode("book");
```

 | **Method:** `checkAll`
`<ejs-treeview id='tree' showCheckBox=true [fields]='field'></ejs-treeview>`
Script

```
@ViewChild("tree") tree: TreeViewComponent;
tree.checkAll("book");
```

 |

| Set checkednodes | **Property:** `checkedNodes`
`<ej-treeview id='tree' showCheckbox=true checkedNodes='checkedNodes' [fields]='field'></ej-treeview>`
Script

```
public checkedNodes=[2, 8, 12]
```

 | **Property:** `checkedNodes`
`<ej-treeview id='tree' checkedNodes='checkedNodes' showCheckBox=true [fields]='field'></ej-treeview>`
Script

```
public checkedNodes=["3", "9", "13"]
```

 |

```
| Get checked nodes | Method: getCheckedNodes  
showCheckbox=true></ej-treeview><br /><br />Script  
$(("#tree").data("ejTreeView"));<br />treeObj.getCheckedNodes(); | Can be achieved  
using,  
<ej-treeview id='tree' showCheckBox=true [fields]='field'></ej-treeview><br /><br />Script  
@ViewChild("tree") tree: TreeViewComponent;<br />var  
check=tree.getAllCheckedNodes();<br />var i=0;<br />var  
checkedNodes;<br />while(i<check.length)  
{<br />checkedNodes.push(tree.element.querySelector("[data-uid=" + checkedNodes[i] +  
""));<br />i++<br />}<br />console.log(checkedNodes) |
```

```
| Get checked nodes index | Method: getCheckedNodesIndex<br/><br/><ej-treeview id='tree'  

[fields]='field' showCheckbox=true</ej-treeview><br /><br/>Script<br/>var treeObj =  

$("#tree").data("ejTreeView");<br />treeObj.getCheckedNodesIndex(); | Can be achieved  

using,<br/><br/><ej-treeview id='tree' showCheckBox=true [fields]='field'></ej-treeview><br  

/><br />Script<br />@ViewChild("tree") tree: TreeViewComponent;<br/>var  

nodes=tree.element.querySelectorAll('e-list-item')<br/>var  

nodeIndex;<br/>while(i<nodes.length) {<br/>if(nodes[i].classList.contains('e-check'))  

{nodeIndex.push(i);<br/>}<br/>i++<br/>}<br/>console.log(nodeIndex) |
```

```
| To check if nodes are checked | Method: isNodeChecked  
[fields]='field' showCheckbox=true></ej-treeview><br /><br />Script<br />var treeObj =  
$("#tree").data("ejTreeView");<br />treeObj.isNodeChecked("book"); | Can be achieved  
using,<br /><br /><ejs-treeview id='tree' showCheckBox=true [fields]='field'></ejs-treeview><br  
<br />Script<br />@ViewChild("tree") tree: TreeViewComponent;<br />if  
(tree.getNode("book")['isChecked'] === true) {<br />console.log("Node is checked")<br /> |
```

```
| Triggers when node is checked successfully | Event: nodeCheck  
showCheckbox=true (nodeCheck)="nodeCheck()" [fields]='field'></ej-treeview><br /><br /></Script><br /></nodeCheck() {} | Event: nodeChecked  
(nodeChecked)="nodeChecked()" showCheckBox=true></ejs-treeview><br /><br /></Script><br /></>public nodeChecked(): void {<br>if (args.action == "check") {<br>|
```

| Checkbox support | **Property:** `showCheckbox`

<ej-treeview id='tree' showCheckbox=true [fields]='field'></ej-treeview> | **Property:** `showCheckBox`

<ejs-treeview id='tree' showCheckBox=true></ejs-treeview> |

```
| To uncheck all nodes | Method: unCheckAll<br/><br/><ej-treeview id='tree' [fields]='field'  
showCheckbox=true></ej-treeview><br /><br/>Script<br/>var treeObj =  
$("#tree").data("ejTreeView");<br />treeObj.unCheckAll(); | Method: uncheckAll<br/><br/><ejs-  
treeview id='tree' showCheckBox=true [fields]='field'></ejs-treeview><br /><br />Script<br  
>@ViewChild("tree") tree: TreeViewComponent;<br/>tree.uncheckAll(); |
```

```
| To uncheck node | Method: uncheckNode<br><br><ej-treeview id='tree' [fields]='field'
showCheckbox=true></ej-treeview><br /><br><b>Script<br>var treeObj =
$("#tree").data("ejTreeView");<br />treeObj.uncheckNode("book"); | Method:
uncheckAll<br><br><ejs-treeview id='tree' showCheckBox=true [fields]='field'></ejs-
```

treeview>

</Script>
@ViewChild("tree") tree:
TreeViewComponent;
tree.uncheckAll(["book"]); |

| Triggers when node is unchecked successfully | **Event:** *nodeUncheck*

<ej-treeview id='tree'
showCheckbox=true (nodeUncheck)="nodeUncheck()" [fields]='field'></ej-treeview>

</Script>
nodeUncheck() {} | **Event:** *nodeChecked*

<ej-treeview id='tree'
(nodeChecked)="nodeChecked()" showCheckbox=true></ej-treeview>

</Script>
public nodeChecked(): void {
if (args.action == "un-check") {}
} |

| Triggers before nodes are checked/ unchecked | Not Applicable | **Event:**
nodeChecking

<ej-treeview id='tree' (nodeChecking)="nodeChecking()"&br/>showCheckbox=true></ej-treeview>

</Script>
public nodeChecking(): void {
if
(args.action == "check") {}
} |

Drag and Drop

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Drag and drop | **Property:** *allowDragAndDrop*

<ej-treeview id='tree'
allowDragAndDrop=true [fields]='field'></ej-treeview> | **Property:**
allowDragAndDrop

<ej-treeview id='tree' allowDragAndDrop=true></ej-treeview> |

| Prevent Drag and drop to another Treeview | **Property:**
allowDragAndDropAcrossControl

<ej-treeview id='tree' allowDragAndDrop=true
allowDragAndDropAcrossControl=true [fields]='field'></ej-treeview> | **Can be achieved using,**

<ej-treeview id='tree' (nodeDragStop)="nodeDragStop()"&br/>allowDragAndDrop=true></ej-treeview>

</Script>
public nodeDragStop(): void
{
if (args.draggedParentNode.closest('.e-treeview') !== args.dropTarget.closest('.e-
treeview')) {}
args.cancel = true;
}} |

| Prevent sibling drop | **Property:** *allowDropSibling*

<ej-treeview id='tree'
allowDragAndDrop=true allowDropSibling=false [fields]='field'></ej-treeview> | **Can be achieved
using,**

<ej-treeview id='tree' allowDragAndDrop=true
(nodeDragStop)="nodeDragStop()" [fields]='field'></ej-treeview>

</Script>

public nodeDragging(): void {
if(args.dropIndicator === "e-drop-next")
{
args.cancel = true;
}} |

| Prevent child drop | **Property:** *allowDropChild*

<ej-treeview id='tree'
allowDragAndDrop=true allowDropChild=false [fields]='field'></ej-treeview> | **Can be achieved
using,**

<ej-treeview id='tree' allowDragAndDrop=true
(nodeDragStop)="nodeDragStop()" [fields]='field'></ej-treeview>

</Script>

public nodeDragging(): void {
if(args.dropIndicator === "e-drop-in")
{
args.cancel = true;
}} |

| Triggers when node is dragged | **Event:** *nodeDrag*

<ej-treeview id='tree'
allowDragAndDrop=true (nodeDrag)="nodeDrag()" [fields]='field'></ej-treeview>

</Script>
nodeDrag() {} | **Event:** *nodeDragging*

<ej-treeview id='tree'

```
(nodeDragging)="nodeDragging()" allowDragAndDrop=true</ej-treeview><br /><br /></Script><br />public nodeDragging(): void {} |
```

| Triggers when node drag is started successfully | **Event:** *nodeDragStart*

<ej-treeview id='tree' allowDragAndDrop=true (nodeDragStart)="nodeDragStart()" [fields]='field'></ej-treeview>

</Script>
nodeDragStart() {} | **Event:** *nodeDragStart*

<ej-treeview id='tree' (nodeDragStart)="nodeDragStart()" allowDragAndDrop=true></ej-treeview>

</Script>
public nodeDragStart(): void {} |

| Triggers before dragged node drag is stopped | **Event:** *nodeDragStop*

<ej-treeview id='tree' allowDragAndDrop=true (nodeDragStop)="nodeDragStop()" [fields]='field'></ej-treeview>

</Script>
nodeDragStop() {} | **Event:** *nodeDragStop*

<ej-treeview id='tree' (nodeDragStop)="nodeDragStop()" allowDragAndDrop=true></ej-treeview>

</Script>
public nodeDragStop(): void {} |

| Triggers when node is dropped successfully | **Event:** *nodeDropped*

<ej-treeview id='tree' allowDragAndDrop=true (nodeDropped)="nodeDropped()" [fields]='field'></ej-treeview>

</Script>
nodeDropped() {} | **Event:** *nodeDropped*

<ej-treeview id='tree' (nodeDropped)="nodeDropped()" allowDragAndDrop=true></ej-treeview>

</Script>
public nodeDropped(): void {} |

Expand/Collapse nodes

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Triggers before node is collapsed | **Event:** *beforeCollapse*

<ej-treeview id='tree' (beforeCollapse)="beforeCollapse()" [fields]='field'></ej-treeview>

</Script>
beforeCollapse() {} | **Event:** *nodeCollapsing*

<ej-treeview id='tree' (nodeCollapsing)="nodeCollapsing()"></ej-treeview>

</Script>
public nodeCollapsing(): void {} |

| Triggers before node is expanded | **Event:** *beforeExpand*

<ej-treeview id='tree' (beforeExpand)="beforeExpand()" [fields]='field'></ej-treeview>

</Script>
beforeExpand() {} | **Event:** *nodeExpanding*

<ej-treeview id='tree' (nodeExpanding)="nodeExpanding()"></ej-treeview>

</Script>
public nodeExpanding(): void {} |

| Collapse all nodes | **Method:** *collapseAll*

<ej-treeview id='tree' [fields]='field'></ej-treeview>

</Script>
var treeObj = \$("#tree").data("ejTreeView");
treeObj.collapseAll(); | **Method:** *collapseAll*

<ej-treeview id='tree' [fields]='field'></ej-treeview>

</Script>
@ViewChild("tree") tree: TreeViewComponent;
tree.collapseAll(); |

| Collapse Node | **Method:** *collapseNode*

<ej-treeview id='tree' [fields]='field'></ej-treeview>

</Script>
var treeObj = \$("#tree").data("ejTreeView");
treeObj.collapseNode("1"); | **Method:** *collapseAll*

<ej-treeview id='tree' [fields]='field'></ej-treeview>

</Script>
@ViewChild("tree") tree: TreeViewComponent;
tree.collapseAll(["1"]); |

| Prevent multiple nodes expand | **Property:** `enableMultipleExpand`
`<ej-treeview id='tree' enableMultipleExpand=false [fields]='field'></ej-treeview>` | **Can be achieved using,**
`<ej-treeview id='tree' (nodeExpanding)="nodeExpanding()" [fields]='field'></ej-treeview>`
`<Script>public nodeExpanding(): void{
 parent=args.node.parentNode.closest('.e-list-item');
 if (parent === null)
 parent=args.node.parentNode.closest('.e-treeview');
 var
 children=parent.querySelectorAll('.e-list-item');
 var i=0;
 var
 nodes=[];
 while(i<children.length){
 nodes.push(children[i].getAttribute("data-
 uid"));
 i++;
 }
 this.collapseAll(nodes)
}` |

| Expand all Nodes | **Method:** `expandAll`
`<ej-treeview id='tree' [fields]='field'></ej-treeview>`
`<Script>var treeObj = $("#tree").data("ejTreeView");
 treeObj.expandAll();` | **Method:** `expandAll`
`<ej-treeview id='tree' [fields]='field'></ej-treeview>`
`<Script>@ViewChild("tree") tree:
 TreeViewComponent;
 tree.expandAll();` |

| Expand Node | **Method:** `expandNode`
`<ej-treeview id='tree' [fields]='field'></ej-treeview>`
`<Script>var treeObj = $("#tree").data("ejTreeView");
 treeObj.expandNode("1");` | **Method:** `expandAll`
`<ej-treeview id='tree' [fields]='field'></ej-treeview>`
`<Script>@ViewChild("tree") tree:
 TreeViewComponent;
 tree.expandAll(["1"]);` |

| Gets/Sets Expanded nodes | **Property:** `expandedNodes`
`<ej-treeview id='tree' [fields]='field' [expandedNodes]='expandedNodes'></ej-treeview>`
`<Script>public
 expandedNodes: string[] = ['1', '2'];` | **Property:** `expandedNodes`
`<ej-treeview id='tree' [expandedNodes]='expandedNodes' [fields]='field'></ej-treeview>`
`<Script>public
 expandedNodes: string[] = ['1', '2'];` |

| Expand action | **Property:** `expandOn`
`<ej-treeview id='tree' [fields]='field' expandOn='click'></ej-treeview>` | **Property:** `expandOn`
`<ej-treeview id='tree' [fields]='field' expandOn='Click'></ej-treeview>` |

| Get expanded nodes | **Method:** `getExpandedNodes`
`<ej-treeview id='tree' [fields]='field'></ej-treeview>`
`<Script>var treeObj =
 $("#tree").data("ejTreeView");
 treeObj.getExpandedNodes();` | **Can be achieved using,**
`<ej-treeview id='tree' [fields]='field'></ej-treeview>`
`<Script>@ViewChild("tree") tree: TreeViewComponent;
 var
 expand=tree.expandedNodes;
 var i=0;
 var
 expandedNodes;
 while(i<expand.length)
 {
 expandedNodes.push(tree.element.querySelector("[data-uid='"+ expandednodes[i] +
 "']"));
 i++;
 }
 console.log(expandedNodes)` |

| Get expanded nodes index | **Method:** `getExpandedNodesIndex`
`<ej-treeview id='tree' [fields]='field'></ej-treeview>`
`<Script>var treeObj =
 $("#tree").data("ejTreeView");
 treeObj.getExpandedNodesIndex();` | Not Applicable |

| To check if node is expanded | **Method:** `isExpanded`
`<ej-treeview id='tree' [fields]='field'></ej-treeview>`
`<Script>var treeObj =`

`$("#tree").data("ejTreeView");
treeObj.isExpanded("book");` | **Can be achieved using,

<ej-treeview id='tree' [fields]='field'></ej-treeview>

<Script
</if(tree.expandedNodes.indexOf("book") !== -1) {
console.log("Node is expanded")
} |**

| Triggers when node is collapsed successfully | **Event:** `nodeCollapse

<ej-treeview id='tree' (nodeCollapse)="nodeCollapse()" [fields]='field'></ej-treeview>

<Script
</nodeCollapse() {}` | **Event:** `nodeCollapsed

<ej-treeview id='tree' (nodeCollapsed)="nodeCollapsed()"></ej-treeview>

<Script
</public nodeCollapsed(): void {}` |

| Triggers when node is expanded successfully | **Event:** `nodeExpand

<ej-treeview id='tree' (nodeExpand)="nodeExpand()" [fields]='field'></ej-treeview>

<Script
</nodeExpand() {}` | **Event:** `nodeExpanded

<ej-treeview id='tree' (nodeExpanded)="nodeExpanded()"></ej-treeview>

<Script
</public nodeExpanded(): void {}` |

Node Editing

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Editing | **Property:** `allowEditing

<ej-treeview id='tree' allowEditing=true [fields]='field'></ej-treeview>` | **Property:** `allowEditing

<ej-treeview id='tree' allowEditing=true [fields]='field'></ej-treeview>` |

| Triggers before node is edited | **Event:** `beforeEdit

<ej-treeview id='tree' (beforeEdit)="beforeEdit()" allowEditing=true [fields]='field'></ej-treeview>

<Script
</beforeEdit() {}` | **Event:** `nodeEditing

<ej-treeview id='tree' (nodeEditing)="nodeEditing()" allowEditing=true></ej-treeview>

<Script
</public nodeEditing(): void {}` |

| To Enable editing programatically | Not Applicable | **Method:** `beginEdit

<ej-treeview id='tree' allowEditing=true [fields]='field'></ej-treeview>

<Script
</@ViewChild("tree") tree: TreeViewComponent;
tree.beginEdit("1");` |

| Triggers before node edit is successful | **Event:** `inlineEditValidation

<ej-treeview id='tree' (inlineEditValidation)="inlineEditValidation()" allowEditing=true [fields]='field'></ej-treeview>

<Script
</inlineEditValidation() {}` | **Event:** `nodeEdited

<ej-treeview id='tree' (nodeEdited)="nodeEdited()" allowEditing=true [fields]='field'></ej-treeview>

<Script
</public nodeEdited(): void {}` |

| Triggers when node is edited successfully | **Event:** `nodeEdit

<ej-treeview id='tree' (nodeEdit)="nodeEdit()" allowEditing=true [fields]='field'></ej-treeview>

<Script
</nodeEdit() {}` | **Event:** `dataSourceChanged

<ej-treeview id='tree' (dataSourceChanged)="dataSourceChanged()" [fields]='field'></ej-treeview>

<Script
</public dataSourceChanged(): void {}` |

Node Selection

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Multi-selection | **Property:** `allowMultiSelection`
`allowMultiSelection=true [fields]='field'></ej-treeview>` | **Property:**
`allowMultiSelection`
`allowMultiSelection=true [fields]='field'></ej-treeview>` |

| Triggers before node is selected | **Event:** `beforeSelect`
`(beforeSelect)="beforeSelect()" [fields]='field'></ej-treeview>`
`

</Script>
beforeSelect() {}` | **Event:** `nodeSelecting`
`(nodeSelecting)="nodeSelecting()" [fields]='field'></ej-treeview>`
`

</Script>
public nodeSelecting(): void {}` |

| Fullrowselection | **Property:** `fullRowSelect`
`fullRowSelect=true [fields]='field'></ej-treeview>` | **Property:** `fullRowSelect`
`fullRowSelect=true [fields]='field'></ej-treeview>` |

| Get selected node | **Method:** `getSelectedNode`
`[fields]='field'></ej-treeview>`
`

</Script>
var treeObj = $(("#tree").data("ejTreeView"));
treeObj.getSelectedNode();` | **Can be achieved using,**
`

</ej-treeview id='tree' [fields]='field'></ej-treeview>`
`

</Script>
@ViewChild("tree") tree: TreeViewComponent;
var select=tree.selectedNodes;
var selectedNode;
selectedNode.push(tree.element.querySelector('[data-uid="' + selectedNode[i] + "']));
console.log(selectedNode)` |

| Get selected node index | **Method:** `getSelectedNodeIndex`
`[fields]='field'></ej-treeview>`
`

</Script>
var treeObj = $(("#tree").data("ejTreeView"));
treeObj.getSelectedNodeIndex();` | **Can be achieved using,**
`

</ej-treeview id='tree' [fields]='field'></ej-treeview>`
`

</Script>
@ViewChild("tree") tree: TreeViewComponent;
var nodes=tree.element.querySelectorAll('.e-list-item')
var nodeIndex;
var i = 0;
while(i<nodes.length) {
if(nodes[i].classList.contains('e-active')) {nodeIndex = i;
break;
}
i++
}
console.log(nodeIndex)` |

| Get selected nodes | **Method:** `getSelectedNodes`
`

</ej-treeview id='tree' [fields]='field' allowMultiSelection=true></ej-treeview>`
`

</Script>
var treeObj = $(("#tree").data("ejTreeView"));
treeObj.getSelectedNodes();` | **Can be achieved using,**
`

</ej-treeview id='tree' allowMultiSelection=true [fields]='field'></ej-treeview>`
`

</Script>
@ViewChild("tree") tree: TreeViewComponent;
var select=tree.selectedNodes;
var i=0;
var selectedNodes;
while(i<select.length) {
selectedNodes.push(tree.element.querySelector('[data-uid="' + selectedNodes[i] + "']));
i++
}
console.log(selectedNodes)` |

| Get selected nodes index | **Method:** `getSelectedNodesIndex`
`[fields]='field' allowMultiSelection=true></ej-treeview>`
`

</Script>
var treeObj = $(("#tree").data("ejTreeView"));
treeObj.getSelectedNodesIndex();` | **Can be achieved using,**
`

</ej-treeview id='tree' allowMultiSelection=true [fields]='field'></ej-treeview>`
`

</Script>
@ViewChild("tree") tree: TreeViewComponent;
var nodes=tree.element.querySelectorAll('.e-list-item')
var nodeIndex;
var i =`

```
0;<br/>while(i<nodes.length) {<br/>if(nodes[i].classList.contains('e-active'))
{nodeIndex.push(i);<br/>}<br/>i++<br/>}<br/>console.log(nodeIndex) |
```

| To check if node is selected | **Method:** *isSelected*

<ej-treeview id='tree'
[fields]='field'></ej-treeview>

Script
var treeObj =
\$("#tree").data("ejTreeView");
treeObj.isSelected("book"); | **Can be achieved**
using,

<ejs-treeview id='tree' [fields]='field'></ejs-treeview>

Script
@ViewChild("tree") tree: TreeViewComponent;
if (tree.selectedNodes.indexOf("book")
!== -1) {
console.log("Node is selected")
} |

| Triggers when node is selected successfully | **Event:** *nodeSelect*

<ej-treeview id='tree'
(nodeSelect)="nodeSelect()" [fields]='field'></ej-treeview>

Script
nodeSelect() {}
| **Event:** *nodeSelected*

<ejs-treeview id='tree' (nodeSelected)="nodeSelected()"
[fields]='field'></ejs-treeview>

Script
public nodeSelected(): void {
if
(args.action == "select") {}
} |

| Select all nodes | **Method:** *selectAll*

<ej-treeview id='tree' [fields]='field'
allowMultiSelection=true></ej-treeview>

Script
var treeObj =
\$("#tree").data("ejTreeView");
treeObj.selectAll(); | **Can be achieved using**,

<ejs-
treeview id='tree' [fields]='field' allowMultiSelection=true></ejs-treeview>

Script
@ViewChild("tree") tree: TreeViewComponent;
var nodes =
tree.element.querySelectorAll('.e-list-item')
var selectednodes;
for(int i = 0; i <
nodes.length; i++) {
selectednodes.push(nodes[i].getAttribute('data-uid'))}
tree.selectedNodes = selectednodes; |

| Gets/Sets selected node | **Property:** *selectedNode*

<ej-treeview id='tree' [fields]='field'
[selectedNode]='selectedNode'></ej-treeview>

Script
public selectedNode:
string[] = ['1']; | **Property:** *selectedNodes*

<ejs-treeview id='tree'
[selectedNodes]='selectedNode' [fields]='field'></ejs-treeview>

Script
public
selectedNode: string[] = ['1']; |

| Select node | **Method:** *selectNode*

<ej-treeview id='tree' [fields]='field'
allowMultiSelection=true></ej-treeview>

Script
var treeObj =
\$("#tree").data("ejTreeView");
treeObj.selectNode(["book"]); | **Can be achieved**
using,

<ejs-treeview id='tree' [fields]='field'></ejs-treeview>

Script
@ViewChild("tree") tree: TreeViewComponent;
tree.selectedNodes=["book"] |

| Gets/Sets selected nodes | **Property:** *selectedNodes*

<ej-treeview id='tree' [fields]='field'
allowMultiSelection=true [selectedNodes]='selectedNodes'></ej-treeview>

Script
public selectedNodes: string[] = ['1', '2']; | **Property:** *selectedNodes*

<ejs-treeview
id='tree' allowMultiSelection=true [selectedNodes]='selectedNodes' [fields]='field'></ejs-
treeview>

Script
public selectedNodes: string[] = ['1', '2']; |

| Triggers when node is unselected successfully | **Event:** *nodeUnselect*

<ej-treeview
id='tree' (nodeUnselect)="nodeUnselect()" [fields]='field'></ej-treeview>

Script
nodeUnselect() {} | **Event:** *nodeSelected*

<ejs-treeview id='tree'
(nodeSelected)="nodeSelected()" [fields]='field'></ejs-treeview>

Script
public
nodeSelected(): void {
if (args.action == "un-select") {}
} |

| To unselect all nodes | **Method:** `unselectAll`
`<ej-treeview id='tree' [fields]='field' allowMultiSelection=true></ej-treeview>

<Script>
var treeObj = $(("#tree").data("ejTreeView"));
treeObj.unselectAll(); | Can be achieved using,

<ejs-treeview id='tree' allowMultiSelection=true [fields]='field'></ejs-treeview>

<Script>
@ViewChild("tree") tree: TreeViewComponent;
tree.selectedNodes=[]; |`

| To unselect node | **Method:** `unselectNode`
`<ej-treeview id='tree' [fields]='field' allowMultiSelection=true></ej-treeview>

<Script>
var treeObj = $(("#tree").data("ejTreeView"));
treeObj.unselectNode("book"); | Can be achieved using,

<ejs-treeview id='tree' allowMultiSelection=true [fields]='field'></ejs-treeview>

<Script>
@ViewChild("tree") tree: TreeViewComponent;
var selectedNodes=tree.selectedNodes.pop(book) |`

Template

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Custom template | **Property:** `template`
`<ej-treeview id='tree' [fields]='field' [template]='templateData'></ej-treeview> | Property: nodeTemplate
<ej-treeview id='tree' [nodeTemplate]='templateData' [fields]='field'></ej-treeview> |`

Uploader

Getting started with Angular Uploader component

This section explains how to create and configure the **uploader** component in Angular. The uploader component is available in `@syncfusion/ej2-angular-inputs` package. Utilize this package to render the uploader Component.

Dependencies

The following packages are required to render the uploader component in your Angular application.

`js

```
|-- @syncfusion/ej2-angular-inputs
|-- @syncfusion/ej2-angular-base
|-- @syncfusion/ej2-angular-popups
|-- @syncfusion/ej2-angular-buttons
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-base
|-- @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-uploader
```

```
,
```

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-uploader --style=scss
```

```
,
```

Navigate to the created project folder.

```
`bash
```

```
cd syncfusion-angular-uploader
```

```
,
```

Installing Syncfusion Uploader 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.

Initialize Uploader

The below couple of steps describes to initialize the uploader component in Angular application.

- Import `UploaderModule` from `@syncfusion/ej2-angular-inputs` package, inject into `imports` section of `NgModule` in the module file `src/app/app.module.ts` to use the `Uploader` component across the application.

```
`typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { UploaderModule } from '@syncfusion/ej2-angular-inputs';
@NgModule({
  imports: [ BrowserModule , UploaderModule],
  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-angular-inputs/styles/material.css';
`
```

The [Custom Resource Generator \(CRG\)](#) is an online web tool, which can be used to generate the custom script and styles for a set of specific components.

This web tool is useful to combine the required component scripts and styles in a single file.

Adding Uploader component

Modify the template in [src/app/app.component.ts] file to render the Uploader component.

Add the Angular Uploader by using `<ejs-uploader>` selector in `template` section of the app.component.ts file.

```
`typescript
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `
    <ejs-uploader #defaultupload [autoUpload]='false'></ejs-uploader>
  `
})
export class AppComponent {
  constructor() {
  }
}
`
```

Running the application

After completing the configuration required to render the uploader, run the application using the following command to display the output in your default browser.

```
ng serve
`
```

From v16.2.41 version, the `Essential JS2 AJAX` library has been integrated for uploader server requests.

Hence, use the third party `promise` library like blue-bird to use the uploader in Internet Explorer.

The below example shows a uploader component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-uploader #defaultupload [autoUpload]='false'></ejs-
uploader>
`
})
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));
```

Adding drop area

By default, the uploader component allows to upload files by drag the files from file explorer, and drop into the drop area.

You can configure any other external element as drop target using `dropArea` property.

In the following sample, external element is configured as drop target to uploader component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='droparea'>
      Drop files here to upload
    </div>
    <div id='uploadfile' >
      <ejs-uploader #defaultupload [autoUpload]='false'
[dropArea]='dropEle'></ejs-uploader>
    </div>
`
})
```



```

    })
    export class AppComponent {
        public dropEle?: HTMLElement ;
        ngOnInit() {
            this.dropEle = document.getElementById('droparea');
        }
        constructor() {
        }
    }
}

```

MAIN.TS

```

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

```

Configure asynchronous settings

By default, the uploader component process the files in Asynchronous mode.

Define the properties `saveUrl` and `removeUrl` to handle the save and remove action as follows.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='droparea'>
      Drop files here to upload
    </div>
    <div id='uploadfile' >
      <ejs-uploader #defaultupload [autoUpload]='false'
[dropArea]='dropEle' [asyncSettings]='path'></ejs-uploader>
    </div>
  `
})
export class AppComponent {
  public dropEle?: any;
  public path: Object = {
    saveUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  constructor() {
  }
}

```

MAIN.TS

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

Handle success and failed upload

You can handle the success and failure actions using the **success** and **failure** events.

To handle these events, define the function and assign it to corresponding event as follows.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-uploader #defaultupload [autoUpload]='false'
    [dropArea]='dropEle' [asyncSettings]='path'
    (success)="onUploadSuccess($event)"
    (failure)="onUploadFailure($event)"></ejs-uploader>
  `
})
export class AppComponent {
  public path: Object = {
    saveUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };

  public onUploadSuccess(args: any): void {
    if (args.operation === 'upload') {
      console.log('File uploaded successfully');
    }
  }

  public onUploadFailure(args: any): void {
    console.log('File failed to upload');
  }

  public dropEle?: HTMLElement ;
  ngOnInit() {
    this.dropEle = document.getElementById('droparea') as HTMLElement;
  }
  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 also explore [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

See Also

- [How to add additional data on upload](#)
- [Achieve file upload programmatically](#)
- [Achieve invisible upload](#)

Note: You can also explore [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Async in Angular Uploader component

The uploader component allows you to upload the files asynchronously.

The upload process requires save and remove action URL to manage the upload process in the server.

- The save action is necessary to handle the upload operation.
- The remove action is optional, one can handle the removed files from server.

The File can be upload automatically or manually. For more information, you can refer to the **Auto Upload** section from the documentation.

Multiple file upload

By Default, the uploader component allows you to select and upload multiple files simultaneously.

The selected files are organized in a list for every file selection until you clear it by clicking clear button that is shown in footer. You can add the multiple attributes to original input element of file by enabling the multiple file selection. The following example explains about **multiple** file upload settings.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-uploader #defaultupload [asyncSettings]='path'></ejs-
    uploader>`
})
```

```

    })
    export class AppComponent {
        public path: Object = {
            saveUrl:
                'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
            removeUrl:
                'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
        };
        constructor() {
        }
    }
}

```

MAIN.TS

```

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

```

Single file upload

You can select and upload a single file by disabling the multiple file selection property.

The file list item is removed for every selection and it always maintain a single file to upload.

You can remove the multiple attributes from the original input element of file by enabling the single file upload property.

The following example explains about single file upload settings.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
    imports: [
        UploaderModule
    ],
    standalone: true,
    selector: 'app-root',
    template: `
        <ejs-uploader #defaultupload [asyncSettings]='path' multiple
        = 'false'></ejs-uploader>
    `
})
export class AppComponent {
    public path: Object = {
        saveUrl:
            'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
        removeUrl:
            'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
    };
    constructor() {
    }
}

```

MAIN.TS

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

Save action

The save action handler upload the files that needs to be specified in the `saveUrl` property.

The save handler receives the submitted files and manages the save process in server.

After uploading the files to server location, the color of the selected file name changes to green and the remove icon is changed as bin icon.

- When the file is uploaded successfully, the event **success** triggers to handle the operation after upload.
- When the file is failed to upload, the event **failure** triggers with information, which cause this failure.

You can cancel the upload process by setting the upload event argument `eventargs.cancel` to true.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-uploader #defaultupload [asyncSettings]='path'></ejs-
uploader>
  `
})
export class AppComponent {
  public path: Object = {
    saveUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Save' };
  constructor() {
  }
}
```

MAIN.TS

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

Server-side configuration for save action

Here's how to handle the server-side action for saving the file in server.

```
`csharp
[AcceptVerbs("Post")]
public void Save()
{
    try
    {
        if (HttpContext.Current.Request.Files.AllKeys.Length > 0)
        {
            var httpPostedFile = HttpContext.Current.Request.Files["UploadFiles"];
            if (httpPostedFile != null)
            {
                var fileSave = HttpContext.Current.Server.MapPath("UploadedFiles");
                var fileSavePath = Path.Combine(fileSave, httpPostedFile.FileName);
                if (!File.Exists(fileSavePath))
                {
                    httpPostedFile.SaveAs(fileSavePath);
                    HttpResponseMessage Response = HttpContext.Current.Response;
                    Response.Clear();
                    Response.ContentType = "application/json; charset=utf-8";
                    Response.StatusDescription = "File uploaded succesfully";
                    Response.End();
                }
            }
            else
            {
                HttpResponseMessage Response = HttpContext.Current.Response;
                Response.Clear();
                Response.Status = "400 File already exists";
                Response.StatusCode = 400;
                Response.StatusDescription = "File already exists";
                Response.End();
            }
        }
    }
}
```

```
}  
}  
}  
}  
catch (Exception e)  
{  
    HttpResponseMessage Response = System.Web.HttpContext.Current.Response;  
    Response.Clear();  
    Response.ContentType = "application/json; charset=utf-8";  
    Response.StatusCode = 400;  
    Response.Status = "400 No Content";  
    Response.StatusDescription = e.Message;  
    Response.End();  
}  
}  
`
```

Server-side configuration for saving and returning responses

The following example demonstrates the server-side action for saving files on the server and returning responses in JSON, String, and File formats.

```
`c#  
[AcceptVerbs("Post")]  
public IActionResult Save()  
{  
    // for JSON Data  
    try  
    {  
        // Process uploaded data  
        var responseData = new  
        {  
            Success = true,  
            Message = "Files uploaded successfully",  
            // Additional data can be added here  
        };  
        return Json(responseData);  
    }  
}
```

```
}  
catch (Exception e)  
{  
    var errorResponse = new  
    {  
        Success = false,  
        Message = "File upload failed: " + e.Message  
    };  
    return Json(errorResponse);  
}  
// for String Data  
try  
{  
    // Process string data  
    var data = "success";  
    // Return the string data  
    return Content(data);  
}  
catch (Exception)  
{  
    var data = "failed";  
    return Content(data);  
}  
// for File Data  
try  
{  
    // Example: Retrieve file path for stream.txt  
    var filePath = "stream.txt"; // Example file path  
    // Get full file path  
    var fullPath = Path.GetFullPath(filePath);  
    // Return the file  
    return PhysicalFile(fullPath, "text/plain");  
}
```



```

catch (Exception e)
{
// Handle file retrieval failure
return Content("Failed to retrieve file response: " + e.Message, "text/plain");
}
}
,

```

Remove action

The remove action is optional. Specify the URL to handle remove process from server. The remove handler receives the posted files and handle the remove operation in server.

- When the files are removed successfully from server, the success event triggers to denote the process has completed.
- When remove action fails, the event **failure** triggers with information, which cause failure in remove process.

You can differentiate the file operation whether the success event triggers from save or remove action in its arguments **eventArgs.operation**.

You can remove the files which is not uploaded locally by clicking the remove icon.

In this case, the success or failure events will not be triggered.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-uploader #defaultupload [asyncSettings]='path'></ejs-
  uploader>
  `
})
export class AppComponent {
  public path: Object = {
    saveUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  constructor() {
  }
}

```

MAIN.TS

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

Server-side configuration for remove action

Here's how to handle the server-side action for removing the file from server.

```
`csharp
[AcceptVerbs("Post")]
public void Remove()
{
    try
    {
        var fileSave = "";
        if (HttpContext.Current.Request.Form["cancelUploading"] != null)
        {
            fileSave = HttpContext.Current.Server.MapPath("UploadingFiles");
        }
        else
        {
            fileSave = HttpContext.Current.Server.MapPath("UploadedFiles");
        }
        var fileName = HttpContext.Current.Request.Files["UploadFiles"].FileName;
        var fileSavePath = Path.Combine(fileSave, fileName);
        if (File.Exists(fileSavePath))
        {
            File.Delete(fileSavePath);
        }
    }
    catch (Exception e)
    {
        HttpResponseMessage Response = HttpContext.Current.Response;
        Response.Clear();
```

```

Response.Status = "404 File not found";
Response.StatusCode = 404;
Response.StatusDescription = "File not found";
Response.End();
}
}
,

```

Auto upload

By default, the uploader processes the files to upload once the files are selected and added in upload queue.

To upload manually, disable the `autoUpload` property.

When you disable this property, you can use the action buttons to call upload all or clear all actions manually.

You can change those buttons text using the `buttons` property in the Uploader component.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-uploader #defaultupload [asyncSettings]='path'
    autoUpload='false'></ejs-uploader>
  `
})
export class AppComponent {
  public path: Object = {
    saveUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  constructor() {
  }
}

```

MAIN.TS

```

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

```

Sequential Upload

By default, the uploader component process multiple files to upload simultaneously. When you enable the [sequentialUpload](#) property, the selected files will process sequentially (one after the other) to the server. If the file uploaded successfully or failed, the next file will upload automatically in this sequential upload. This feature helps to reduce the upload traffic and reduce the failure of file upload.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-uploader #defaultupload [asyncSettings]='path'
    sequentialUpload='true'></ejs-uploader>
  `
})
export class AppComponent {
  public path: Object = {
    saveUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  constructor() {
  }
}
```

MAIN.TS

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

Preloaded files

The uploader component allows you to preloaded the list of files that are uploaded in the server.

The preloaded files are useful to view and remove the files from server that can be achieved by the **files** property.

By default, the files are configured with uploaded successfully state on rendering file list.

The following properties are mandatory to configure the preloaded files:

- Name
- Size

- Type

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-uploader #defaultupload [asyncSettings]='path'
    [files]='preLoadFiles'></ejs-uploader>
  `
})
export class AppComponent {
  public path: Object = {
    saveUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  public preLoadFiles: Object[] = [
    {name: 'Books', size: 500, type: '.png'},
    {name: 'Movies', size: 12000, type: '.pdf'},
    {name: 'Study materials', size: 500000, type: '.docx'},
  ]
  constructor() {
  }
}
```

MAIN.TS

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

Adding additional HTTP headers with upload action

The Uploader component allows you to add the additional headers with **save** and **remove** action request using **uploading** and **removing** event, which helps to send validation token on file upload. Access the current request and set the request header within these events.

The following code block shows how to add the additional headers with save and remove action request.

```
`typescript
```

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

@Component({
```

```
selector: 'app-root',
template: `
<ejs-uploader #defaultupload [asyncSettings]='path' autoUpload='false' (uploading) =
"addHeaders($event)" (removing) = "addHeaders($event)"></ejs-uploader>
`
})
export class AppComponent {
  public path: Object = {
    saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
    removeUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Remove' };
  constructor() {
  }
  public addHeaders(args: any) {
    args.currentRequest.setRequestHeader('custom-header', 'Syncfusion');
  }
}
`
```

You can also explore [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

See Also

- [How to add additional data on upload](#)
- [How to add confirm dialog to remove the files](#)
- [Check the MIME type of file before uploading it](#)
- [How to open and edit the uploaded files](#)

Note: [View Server Side Sample in GitHub](#).

Chunk upload in Angular Uploader component

The Uploader sends the large file split into small chunks and transmits to the server using AJAX. You can also pause, resume, and retry the failed chunk file.

* The chunk upload works in asynchronous upload only.

- This feature is available from the Essential Studio Vol 2, 2018 release.

To enable the chunk upload, set the size to [chunkSize](#) option of the upload and it receives the value in bytes.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./index.css'],
  template: `<div class="chunkupload">
    <ejs-uploader #defaultupload [asyncSettings]='path'></ejs-
uploader>
    </div>
`
})
export class AppComponent {
  public path: Object = {
    saveUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Remove',
    // set chunk size for enable the chunk upload
    chunkSize: 102400
  };
  constructor() {
  }
}

```

MAIN.TS

```

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);

```

The chunk upload functionality separates the selected files into blobs of the data or chunks. These chunks are transmitted to the server using an AJAX request.

The chunks are sent in **sequential** order, and the next chunk can be sent to the server according to the [success](#) of the previous chunk. If any one of the chunk failed, then the remaining chunk cannot be sent to the server.

The [chunkSuccess](#) or [chunkFailure](#) event will be triggered when the chunk is sent to the server successfully or failed. If all the chunks are sent to the server successfully, the uploader success event is triggered.

Chunk upload will work when the selected file size is greater than the specified chunk size. otherwise, it upload the files normally.

Additional configurations

To modify the chunk upload, the following options can be used.

- **RetryAfterDelay** - If error occurs while sending any chunk request from JavaScript, hold the operation for 500 milliseconds (by default), and retry the operation using chunk. This can be achieved by using the [asyncSettings.retryAfterDelay](#) property. You can modify the holding time interval in milliseconds.
- **RetryCount** - Specifies the number of retry actions performed when the file fails to upload. By default, [retry](#) action is performed 3 times. If the file fails to upload continuously, the request is aborted and the uploader [failure](#) event will trigger.

The following sample specifies the chunk upload delay with 3000 milliseconds and the retry count is 5. The failure event is triggered as the wrong saveUrl is used.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./index.css'],
  template: `<div class="chunkupload">
    <ejs-uploader #defaultupload [asyncSettings]='path'></ejs-
uploader>
  </div>
`
})
export class AppComponent {
  public path: Object = {
    // provided the wrong url to showcase the chunk upload failure related
    // properties.
    saveUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Remove',
    // set chunk size for enable the chunk upload
    chunkSize: 102400,
    // set count for automatic retry when chunk upload failed
    retryCount: 5,
    // set time delay for automatic retry when chunk upload failed
    retryAfterDelay: 3000
  };
  constructor() {
  }
}
```

MAIN.TS

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


Resumable upload

Allows you to resume an upload operation after a network failure or manually interrupts (pause) the upload. You can perform pause and resume upload actions using public methods ([pause](#) and [resume](#)) and UI interaction. The pause icon is enabled after the upload begins.

This pause and resume features available only when the chunk upload is enabled.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./index.css'],
  template: `<div class="chunkupload">
    <ejs-uploader #defaultupload [asyncSettings]='path'></ejs-
uploader>
    </div>
`
})
export class AppComponent {
  public path: Object = {
    saveUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Remove',
    // set chunk size for enable the chunk upload
    chunkSize: 102400
  };
  constructor() {
  }
}
```

MAIN.TS

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

Cancel upload

The uploader component allows you to cancel the uploading file. This can be achieved by clicking the cancel icon or using the [cancel](#) method. The [cancelling](#) event will be fired whenever the file upload request is canceled. While canceling the upload request, the partially uploaded file is removed from the server.

When the request fails, the pause icon is changed to retry icon. By clicking the retry icon, sends the failed chunk request again to the server and upload started from where it is failed. You can retry the canceled upload request again using retry UI or [retry](#) methods. But, if you retry this, the file upload action again starts from initial.

The following example explains about chunk upload with cancel support.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  styleUrls: ['./index.css'],
  template: `<div class="chunkupload">
    <ejs-uploader #defaultupload [asyncSettings]='path'></ejs-
  uploader>
    </div>
`
})
export class AppComponent {
  public path: Object = {
    saveUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Remove',
    // set chunk size for enable the chunk upload
    chunkSize: 102400
  };
  constructor() {
  }
}
```

MAIN.TS

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

The retry action has different working behavior for chunk upload and default upload.

- Chunk upload - Retries to upload the failed request where it is failed previously.
- Default upload - Retries to upload the failed file again from initial.

Server-Side configurations

The server-side implementation entirely depends on the application requirements and logic. The following code snippet provides the server-side logic to handle the chunk upload using the uploader components.

```
`csharp
```

```
// Server configuration for upload a file.
```

```
public void Save()
```

```
{
try
{
if (HttpContext.Current.Request.Files.AllKeys.Length > 0)
{
var httpPostedChunkFile = HttpContext.Current.Request.Files["chunkFile"];
if (httpPostedChunkFile != null)
{
var saveFile = HttpContext.Current.Server.MapPath("UploadingFiles");
// Save the chunk file in temporary location with .part extension
var SaveFilePath = Path.Combine(saveFile, httpPostedChunkFile.FileName + ".part");
var chunkIndex = HttpContext.Current.Request.Form["chunkIndex"];
if (chunkIndex == "0")
{
httpPostedChunkFile.SaveAs(SaveFilePath);
}
else
{
// Merge the current chunk file with previous uploaded chunk files
MergeChunkFile(SaveFilePath, httpPostedChunkFile.InputStream);
var totalChunk = HttpContext.Current.Request.Form["totalChunk"];
if (Convert.ToInt32(chunkIndex) == (Convert.ToInt32(totalChunk) - 1))
{
var savedFile = HttpContext.Current.Server.MapPath("UploadedFiles");
var originalFilePath = Path.Combine(savedFile, httpPostedChunkFile.FileName);
// After all the chunk files completely uploaded, remove the .part extension and move this file into save
location
System.IO.File.Move(SaveFilePath, originalFilePath);
}
}
HttpResponse ChunkResponse = HttpContext.Current.Response;
ChunkResponse.Clear();
ChunkResponse.ContentType = "application/json; charset=utf-8";
```

```
ChunkResponse.StatusDescription = "File uploaded succesfully";
ChunkResponse.End();
}
var httpPostedFile = HttpContext.Current.Request.Files["UploadFiles"];
if (httpPostedFile != null)
{
    var fileSave = HttpContext.Current.Server.MapPath("UploadedFiles");
    var fileSavePath = Path.Combine(fileSave, httpPostedFile.FileName);
    if (!File.Exists(fileSavePath))
    {
        httpPostedFile.SaveAs(fileSavePath);
        HttpResponse Response = HttpContext.Current.Response;
        Response.Clear();
        Response.ContentType = "application/json; charset=utf-8";
        Response.StatusDescription = "File uploaded succesfully";
        Response.End();
    }
    else
    {
        HttpResponse Response = HttpContext.Current.Response;
        Response.Clear();
        Response.Status = "400 File already exists";
        Response.StatusCode = 400;
        Response.StatusDescription = "File already exists";
        Response.End();
    }
}
}
}
catch (Exception e)
{
    HttpResponse Response = HttpContext.Current.Response;
    Response.Clear();
```

```
Response.ContentType = "application/json; charset=utf-8";
Response.StatusCode = 400;
Response.Status = "400 No Content";
Response.StatusDescription = e.Message;
Response.End();
}
}
// Server configuration for remove a uploaded file
public void Remove()
{
    try
    {
        var fileSave = "";
        if (HttpContext.Current.Request.Form["cancelUploading"] != null)
        {
            fileSave = HttpContext.Current.Server.MapPath("UploadingFiles");
        } else
        {
            fileSave = HttpContext.Current.Server.MapPath("UploadedFiles");
        }
        var fileName = HttpContext.Current.Request.Files["UploadFiles"].FileName;
        var fileSavePath = Path.Combine(fileSave, fileName);
        if (File.Exists(fileSavePath))
        {
            File.Delete(fileSavePath);
        }
    }
    catch (Exception e)
    {
        HttpResponse Response = HttpContext.Current.Response;
        Response.Clear();
        Response.Status = "404 File not found";
        Response.StatusCode = 404;
    }
}
```

```
Response.StatusDescription = "File not found";
Response.End();
}
}
// Merge the current chunk file with previous uploaded chunk files
public void MergeChunkFile(string fullPath, Stream chunkContent)
{
    try
    {
        using (FileStream stream = new FileStream(fullPath, FileMode.Append, FileAccess.Write,
        FileShare.ReadWrite))
        {
            using (chunkContent)
            {
                chunkContent.CopyTo(stream);
            }
        }
    }
    catch (IOException ex)
    {
        throw ex;
    }
}
```

You can also explore [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

File source in Angular Uploader component

Paste to upload

The uploader component allows you to upload the files using the select or drop files option from the file explorer. It also supports pasting to upload the image files. You can upload any currently copied images in the clipboard.

When you paste the image, it will be saved in the server with the filename as `image.png`. The file name can be renamed in the server end. You can generate a random name for the file name using `getUniqueID` method.

Refer to the following example.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
import { UploadingEventArgs } from '@syncfusion/ej2-inputs';
import { getUniqueID } from '@syncfusion/ej2-base';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-uploader #defaultupload [asyncSettings]='path'
    autoUpload='false' (uploading)='onUploadBegin($event)'></ejs-uploader>
  `
})
export class AppComponent {
  public path: Object = {
    saveUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  constructor() {
  }
  public onUploadBegin(args: UploadingEventArgs) {
    // check whether the file is uploading from paste.
    if (args.fileData.fileSource === 'paste') {
      let newName: string = getUniqueID(args.fileData.name.substring(0,
args.fileData.name.lastIndexOf('.'))) + '.png';
      args.customFormData = [{ 'fileName': newName }];
    }
  }
}
```

MAIN.TS

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

Save action for paste to upload

`csharp

```
public void Save() {
```

```
var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];
```

```
var fileSave = System.Web.HttpContext.Current.Server.MapPath("UploadedFiles");
```

```
var fileSavePath = Path.Combine(fileSave, httpPostedFile.FileName);
```

```

if (!System.IO.File.Exists(fileSavePath)) {
    httpPostedFile.SaveAs(fileSavePath);
    var newName = System.Web.HttpContext.Current.Request.Form["fileName"];
    var filePath = Path.Combine(fileSavePath.Substring(0, fileSavePath.LastIndexOf("/")), newName);
    // Rename the file
    System.IO.File.Move(fileSavePath, newName);
    HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
    Response.Clear();
    Response.ContentType = "application/json; charset=utf-8";
    Response.StatusDescription = fileSavePath;
    Response.End();
}
}
,

```

Directory upload

The uploader component allows you to upload all files in the folders to server by using the [directoryUpload](#) property. When this property is enabled, the uploader component processes the files by iterating through the files and sub-directories in a directory.

It allows you to select only folders instead of files to upload.

The directory upload is available only in browsers that supports **HTML5 directory**. The uploader will process directory upload by dragging and dropping in the Edge browser.

Refer to the following example to upload files to the server.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-uploader #defaultupload [asyncSettings]='path'
    directoryUpload='true'></ejs-uploader>
  `
})
export class AppComponent {
  public path: Object = {
    saveUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Save',

```



```
        removeUrl:
        'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
    };
    constructor() {
    }
}
```

MAIN.TS

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

Save action for directory upload

```
`csharp
public void Save() {
var httpPostedFile = HttpContext.Current.Request.Files["UploadFiles"];
var fileSave = HttpContext.Current.Server.MapPath("UploadedFiles");
// split the folders by using file name
string[] folders = httpPostedFile.FileName.Split('/');
string fileSavePath = "";
if (folders.Length > 1)
{
for (var i = 0; i < folders.Length - 1; i++)
{
var newFolder = Path.Combine(fileSave, folders[i]);
// create folder
Directory.CreateDirectory(newFolder);
fileSave = newFolder;
}
fileSavePath = Path.Combine(fileSave, folders[folders.Length - 1]);
}
else
{
fileSavePath = Path.Combine(fileSave, httpPostedFile.FileName);
}
if (!System.IO.File.Exists(fileSavePath))
{
```

```
// save file in the corresponding server location
httpPostedFile.SaveAs(fileSavePath);
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
// Sending the file path to client side
Response.StatusDescription = fileSavePath;
Response.End();
}
}
,
```

Drag and drop

The uploader component allows you to drag and drop the files to upload.

You can drag the files from file explorer and drop into the drop area.

By default, the Uploader component act as drop area element.

The drop area gets highlighted when you drag the files over drop area.

Custom drop area

The uploader component allows you to set external target element as drop area using the `dropArea` property.

The element can be represented as HTML element or element's ID.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='droparea'> Drop files here to upload </div>
    <ejs-uploader #defaultupload [asyncSettings]='path'
    autoUpload='false' [dropArea]='ele'></ejs-uploader>
  `
})
export class AppComponent {
  public ele?: HTMLElement;
  public path: Object = {
    saveUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
```

```

        removeUrl:
        'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
    };
    ngOnInit() {
        this.ele = document.getElementById('droparea') as HTMLElement;
    }
    constructor() {
    }
}

```

MAIN.TS

```

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

```

Customize drop area

You can customize the appearance of drop area by overriding the default drop area styles.

The class **e-upload-drag-hover** is available to handle this customization.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { UploaderModule } from '@syncfusion/ej2-angular-inputs';
import { Component } from '@angular/core';
@Component({
    imports: [
        UploaderModule
    ],
    standalone: true,
    selector: 'app-root',
    template: `
        <div id='dropArea' style='height: auto; overflow: auto'>
            <span id='drop'> Drop files here or <a href=""
id='browse'><u>Browse</u></a> </span>
            <ejs-uploader #defaultupload [asyncSettings]='path'
[dropArea]='ele'></ejs-uploader>
        </div>
    `
})
export class AppComponent {
    public path: Object = {
        saveUrl:
        'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
        removeUrl:
        'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
    };
    public ele?: HTMLElement;
    ngOnInit() {
        this.ele = document.getElementById('dropArea') as HTMLElement;
        (document.getElementById('browse') as HTMLElement).onclick =
function() {

```

```

        (document.getElementsByClassName('e-file-select-wrap')[0].querySelector('button') as HTMLButtonElement).click();
        return false;
    }

    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 also explore [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

See Also

- [Achieve file upload programmatically](#)
- [Validate image/* on drop](#)

Validation in Angular Uploader component

The uploader component validate the selected files extension and size using the `allowedExtensions`, `minFileSize` and `maxFileSize` properties. The files can be validated before uploading to the server and can be ignored on uploading. Also, you can validate the files by setting the HTML attributes to the original input element. The validation process occurs on drag-and-drop the files also.

File type

You can allow the specific types of files alone to upload using the `allowedExtensions` property. The extension can be represented as collection by comma separators. The uploader component filters the selected or dropped files matched against the specified file types and processes the upload operation. The validation happens when you specify value to inline attribute accept to original input element.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-uploader #defaultupload [asyncSettings]='path'
    allowedExtensions = '.jpg,.png'></ejs-uploader>
  `
})

```

```

    })
    export class AppComponent {
        public path: Object = {
            saveUrl:
                'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
            removeUrl:
                'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
        };
        constructor() {
        }
    }
}

```

MAIN.TS

```

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

```

File size

The uploader component allows you to validate the files based on its size. The validation helps to restrict uploading large files or empty files to the server. The size can be represented in **bytes**. By default, the uploader component allows you to upload **minimum file size** as 0 byte and **maximum file size** as 28.4 MB using **minFileSize** and **maxFileSize** properties.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { UploaderModule } from '@syncfusion/ej2-angular-inputs';
import { Component } from '@angular/core';
@Component({
    imports: [
        UploaderModule
    ],
    standalone: true,
    selector: 'app-root',
    template: `
        <ejs-uploader #defaultupload [asyncSettings]='path'
        minFileSize = 10000 maxFileSize = 1500000></ejs-uploader>
    `
})
export class AppComponent {
    public path: Object = {
        saveUrl:
            'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
        removeUrl:
            'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
    };
    constructor() {
    }
}

```

MAIN.TS

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

Maximum files count

You can restrict uploading the maximum number of files using the **selected** event. In the selected event arguments, you can get the currently selected files details using the `getFilesData()`. You can modify the files details and assign the modified file list to the `eventArgs.modifiedFilesData`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component, ViewChild } from '@angular/core';
import { FileInfo, SelectedEventArgs } from '@syncfusion/ej2-inputs';
import { detach } from '@syncfusion/ej2-base';
import { createSpinner, showSpinner, hideSpinner } from '@syncfusion/ej2-
popups';
import { UploaderComponent } from '@syncfusion/ej2-angular-inputs';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-uploader #defaultupload autoUpload='false'
[asyncSettings]='path' minFileSize = 10000 allowedExtensions = '.doc, .docx,
.xls, .xlsx' (selected)="onFileSelected($event)"
(success)="onUploadSuccess($event)"></ejs-uploader>
  `
})
export class AppComponent {
  public path: Object = {
    saveUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  @ViewChild('defaultupload')
  public uploadObj?: UploaderComponent;
  public onFileSelected(args : SelectedEventArgs) : void {
    // Filter the 5 files only to showcase
    args.filesData.splice(5);
    let filesData : FileInfo[] = (this.uploadObj as
UploaderComponent).getFilesData();
    let allFiles : FileInfo[] = filesData.concat(args.filesData);
    if (allFiles.length > 5) {
      for (let i : number = 0; i < allFiles.length; i++) {
        if (allFiles.length > 5) {
          allFiles.shift();
        }
      }
    }
    args.filesData = allFiles;
  }
}
```

```

        // set the modified custom data
        args.modifiedFilesData = args.filesData;
    }
    args.isModified = true;
}
public onUploadSuccess(args: any): void {
    let li: HTMLElement = (this.uploadObj as
any).uploadWrapper.querySelector('[data-file-name="' + args.file.name +
'"]');
    if (args.operation === 'upload') {
        (li.querySelector('.e-file-delete-btn') as HTMLElement).onclick =
() => {
            this.generateSpinner((this.uploadObj as any).uploadWrapper);
        };
        (li.querySelector('.e-file-delete-btn') as HTMLElement).onkeydown
= (e: any) => {
            if (e.keyCode === 13) {
                this.generateSpinner(e.target.closest('.e-upload'));
            }
        };
    } else {
        hideSpinner((this.uploadObj as any).uploadWrapper);
        detach((this.uploadObj as any).uploadWrapper.querySelector('.e-
spinner-pane'));
    }
}
public generateSpinner(targetElement: HTMLElement): void {
    createSpinner({ target: targetElement, width: '25px' });
    showSpinner(targetElement);
}
constructor() {
}
}

```

MAIN.TS

```

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

```

Duplicate files

You can validate the duplicate files before uploading to server using the selected event. Compare the selected files with the existing files data and filter the file list by removing the duplicate files.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component, ViewChild } from '@angular/core';
import { FileInfo } from '@syncfusion/ej2-inputs';
import { isNullOrUndefined } from '@syncfusion/ej2-base';
import { UploaderComponent } from '@syncfusion/ej2-angular-inputs';
@Component({

```

```

imports: [
    UploaderModule
],
standalone: true,
selector: 'app-root',
template: `
    <ejs-uploader #defaultupload autoUpload='false'
[asyncSettings]='path' (selected)="onFileSelect($event)" ></ejs-uploader>
`
))
export class AppComponent {
    public path: Object = {
        saveUrl:
            'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
        removeUrl:
            'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
    };
    @ViewChild('defaultupload')
    public uploadObj?: UploaderComponent;
    public onFileSelect(args : any) {
        let existingFiles: FileInfo[] = (this.uploadObj as any).getFilesData();
        for (let i: number = 0; i < args.filesData.length; i++) {
            for(let j: number = 0; j < existingFiles.length; j++) {
                if (!isNullOrUndefined(args.filesData[i])) {
                    if (existingFiles[j].name == args.filesData[i].name) {
                        args.filesData.splice(i, 1);
                    }
                }
            }
        }
        existingFiles = existingFiles.concat(args.filesData);
        args.modifiedFilesData = existingFiles;
        args.isModified = 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));

```

You can also explore [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

See Also

- [Validate image/* on drop](#)
- [Determine whether uploader has file input \(required validation\)](#)
- [Check file size before uploading it](#)

- [Check the MIME type of file before uploading it](#)

Form support in Angular Uploader component

The uploader component works with HTML form like default file input.

The following configuration is must to make the uploader work inside the form.

- `saveUrl` and `removeUrl` must be null.
- `autoUpload` must be disabled.
- `name` attribute must be added in input element.

The selected or dropped files are received as a collection in form action when the form is submitted.

The form action handles the server-side operations that manage the file upload process.

When you reset the form, the file list and data will be cleared.

APP.COMPONENT.TS

```
import { BrowserModule } from '@angular/platform-browser'
import { NgModule, ModuleWithProviders, CUSTOM_ELEMENTS_SCHEMA } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { Component, ViewChild } from '@angular/core';
import { EmitType } from '@syncfusion/ej2-base';
import { Dialog } from '@syncfusion/ej2-popups';
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
import { DialogComponent } from '@syncfusion/ej2-angular-popups';
/**
 * Default Uploader Default Component
 */
@Component({
  imports: [
    UploaderModule, DialogModule, FormsModule,
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './formsupport.html',
  styleUrls: ['./formsupport.css']
})
export class AppComponent {
  @ViewChild('Dialog')
  public Dialog?: DialogComponent;
  public width: string = '335px';
  public visible: boolean = false;
  public content: string = 'Your details has been updated successfully, Thank you';
  public target: string = '#control_wrapper';
  public isModal: boolean = true;
  public animationSettings: object = {
    effect: 'Zoom'
  }
  public options: object = {
    rules: {
```

```

        'name': {
            required: true
        },
        'email': {
            required: true
        },
        'upload': {
            required: true
        }
    }
    @ViewChild('formElement') element: any;
    ngAfterViewInit() {
        let formObject: FormValidator = new
FormValidator(this.element.nativeElement, this.options);
        // validate all input elements in the form
    }
    browseClick() {
        (document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button') as HTMLButtonElement).click(); return
false;
    }
    Submit() {
        this.onFormSubmit();
    }
    public onFileSelect: EmitType<Object> = (args: any) => {
        let inputElement: HTMLInputElement =
document.getElementById('upload') as HTMLInputElement;
        inputElement.value = args.filesData[0].name;
    }
    // Close the modal Dialog on overlay click
    public overlayClick(): void {
        this.Dialog?.hide();
    }
    public onFormSubmit(): void {
        let formObject: FormValidator = new FormValidator("#form1",
this.options);
        let formStatus: Boolean = formObject.validate();
        if (formStatus) {
            formObject.element.reset();
            this.Dialog?.show();
        }
    }
}

```

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

By using **ngModel** directive, you can bind the model to the uploader in template-driven forms.

For more details, refer to the [Angular Documentation](#)

The following sample demonstrates how to render uploader component with required validation inside the template-driven forms.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component, OnInit, ViewChild } from '@angular/core';
import { DialogComponent } from '@syncfusion/ej2-angular-popups';
import { EmitType } from '@syncfusion/ej2-base';

@Component({
  imports: [ FormsModule, DialogModule, UploaderModule, ButtonModule,
    ReactiveFormsModule ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="control-section">
    <div class="col-lg-12">
      <h4 class="form-title">Photo Contest</h4>
      <div class="control_wrapper" id="control_wrapper" style="margin: 10px
auto;">
        <form id="template_driven" #userForm="ngForm" novalidate>
          <div class="form-group" style="padding-top: 11px;">
            <div class="e-float-input">
              <input type="text" id="name" #nameval='ngModel'
name="name" required ngModel>
              <span class="e-float-line"></span>
              <label class="e-float-text e-label-top"
for="name">Name</label>
              <div *ngIf="(nameval.invalid && (nameval.dirty ||
nameval.touched))">
                <div class="e-error" *ngIf="nameval.errors">
                  * Enter your name
                </div>
              </div>
            </div>
            <div class="form-group" style="padding-top: 11px;">
              <div class="e-float-input upload-area">
                <input type="text" id="upload" #uploadval='ngModel'
[(ngModel)]="uploadInput" readonly name="upload" required ngModel>
                <button id="browse" class="e-control e-btn e-info"
(click)='browseClick()'>Browse...</button>
                <span class="e-float-line"></span>
                <label class="e-float-text e-label-top"
for="upload">Choose a file</label>
                <div *ngIf="(uploadval.invalid && (uploadval.dirty
|| uploadval.touched))">
                  <div class="e-error" *ngIf="uploadval.errors">
                    * Select a file
                  </div>
                </div>
              </div>
            </div>
          </div>
        </form>
      </div>
    </div>
  `
})
```

```

        </div>
        <ejs-uploader #defaultupload id='fileupload'
allowedExtensions="image/*" [autoUpload]=false [multiple]='multiple'
(selected)='onFileSelect($event)'></ejs-uploader>
    </div>
    <div class="form-group" style="padding-top: 11px;">
        <div class="submitBtn">
            <button class="submit-btn e-btn" id="submit-btn"
[disabled]="userForm.invalid" type="reset" (click)=
"Submit()">Submit</button>
            <div class="desc"><span>*This button is not a submit
type and the form submit handled from externally.</span></div>
        </div>
    </div>
</form>
    <ejs-dialog id="confirmationDialog" #Dialog [buttons]='dlgButtons'
[animationSettings]='animationSettings' [header]='formHeader'
[showCloseIcon]='showCloseIcon' [content]='content' [target]='target'
[width]='width' [visible]="visible" [isModal]="isModal" >
    </ejs-dialog>
</div>
</div>
</div>`
})
export class AppComponent {
    @ViewChild('Dialog')
    public dialogObj?: DialogComponent;
    public width: string = '335px';
    public visible: boolean = false;
    public multiple: boolean = false;
    public showCloseIcon: Boolean = true;
    public formHeader: string = 'Success';
    public content: string = 'Your details have been updated successfully,
Thank you.';
    public target: string = '#control_wrapper';
    public isModal: boolean = true;
    public animationSettings: object = {
        effect: 'Zoom'
    };
    public uploadInput: string = '';
    public dlgBtnClick: EmitType<object> = () => {
        (this.dialogObj as DialogComponent).hide();
    }
    public dlgButtons: Object[] = [{ click: this.dlgBtnClick.bind(this),
buttonModel: { content: 'Ok', isPrimary: true } }];
    @ViewChild('formElement') element: any;
    public browseClick() {
        (document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button') as HTMLButtonElement).click(); return
false;
    }
    public Submit(): void {
        this.onFormSubmit();
    }
    public onFileSelect: EmitType<Object> = (args: any) => {
        this.uploadInput = args.filesData[0].name;
    }
}

```

```
public onFormSubmit(): void {
    (this.dialogObj 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));
```

Reactive forms

You can render the uploader component inside the reactive forms.

The reactive forms rendered with the help of **FormGroup**.

For more details, refer to the [Angular Documentation](#)

The following sample demonstrates how to render uploader component with required validation inside the **reactive forms**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { Component, Inject, OnInit, ViewChild } from '@angular/core';
import { FormGroup, FormBuilder, Validators, FormControl } from
'@angular/forms';
import { EmitType } from '@syncfusion/ej2-base';
import { DialogComponent } from '@syncfusion/ej2-angular-popups';
@Component({
    imports: [ FormsModule, DialogModule, UploaderModule, ButtonModule,
        ReactiveFormsModule ],
    standalone: true,
    selector: 'app-root',
    template: `<div class="control-section">
        <div class="col-lg-12">
            <h4 class="form-title">Photo Contest</h4>
            <div class="control_wrapper" id="control_wrapper" style="margin: 25px
auto;">
                <form id="reactive" [formGroup]="form">
                    <div class="form-group" style="padding-top: 11px;">
                        <div class="e-float-input">
                            <input type="text" id="name" name="name"
class="required" formControlName="name">
                                <span class="e-float-line"></span>
                                <label class="e-float-text e-label-top"
for="name">Name</label>
                        </div>
                        <app-field-error-display
[displayError]="isFieldValid('name')" errorMsg="* Please Enter your name">
```

```

        </app-field-error-display>
    </div>
    <div class="form-group" style="padding-top: 11px;">
        <div class="e-float-input upload-area">
            <input type="text" id="upload" name="upload"
[(ngModel)]="uploadInput" readonly FormControlName="upload" class="required">
            <button id="browse" class="e-control e-btn e-info"
(click)='browseClick()'>Browse...</button>
            <span class="e-float-line"></span>
            <label class="e-float-text e-label-top"
for="upload">Choose a file</label>
        </div>
        <app-field-error-display
[displayError]="isFieldValid('upload')" errorMsg="* Select any file">
        </app-field-error-display>
        <ejs-uploader #defaultupload id='fileupload'
allowedExtensions="image/*" [autoUpload]=false [multiple]='multiple'
(selected)='onFileSelect($event)'></ejs-uploader>
    </div>
    <div class="form-group" style="padding-top: 11px;">
        <div class="submitBtn">
            <button class="submit-btn e-btn" id="submit-btn"
[disabled]="form.invalid" (click)="Submit()">Submit</button>
            <div class="desc"><span>*This button is not a submit type
and the form submit handled from externally.</span></div>
        </div>
    </div>
</form>
    <ejs-dialog id="confirmationDialog" #Dialog [buttons]='dlgButtons'
[animationSettings]='animationSettings' [header]='formHeader'
[showCloseIcon]='showCloseIcon' [content]='content' [target]='target'
[width]='width' [visible]="visible" [isModal]="isModal" >
    </ejs-dialog>
</div>
</div>
</div>`
    })
    export class AppComponent {
        @ViewChild('Dialog')
        public dialogObj?: DialogComponent;
        public form?: FormGroup | any;
        public width: string = '335px';
        public visible: boolean = false;
        public multiple: boolean = false;
        public showCloseIcon: Boolean = true;
        public formHeader: string = 'Success';
        public content: string = 'Your details have been updated successfully,
Thank you.';
        public target: string = '#control_wrapper';
        public isModal: boolean = true;
        public animationSettings: any = {
            effect: 'Zoom'
        };
        private formSumitAttempt?: boolean;
        public dlgBtnClick: EmitType<object> = () => {
            this.dialogObj?.hide();
        }
    }

```

```

    public dlgButtons: Object[] = [{ click: this.dlgBtnClick.bind(this),
buttonModel: { content: 'Ok', isPrimary: true } }];
    public uploadInput: string = '';
    public browseClick() {
        (document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button') as HTMLButtonElement).click(); return
false;
    }
    public Submit(): void {
        this.onFormSubmit();
    }
    public onFileSelect: EmitType<Object> = (args: any) => {
        this.uploadInput = args.filesData[0].name;
    }
    public onFormSubmit(): void {
        this.formSumitAttempt = true;
        if (this.form?.valid) {
            (this.dialogObj as DialogComponent).show();
            this.form.reset();
        } else {
            this.validateAllFormFields(this.form as any);
        }
    }
    constructor(@Inject(FormBuilder) public formBuilder: FormBuilder) {}
    ngOnInit() {
        this.form = this.formBuilder.group({
            name: [null, Validators.required],
            upload: [null, Validators.required],
        });
    }
    isFieldValid(field: string) {
        return (!((this.form as any).get(field).valid && (this.form as
any).get(field).touched) ||
            ((this.form as any).get(field).untouched && this.formSumitAttempt));
    }
    validateAllFormFields(formGroup: FormGroup) {
        Object.keys(formGroup.controls).forEach(field => {
            const control = formGroup.get(field);
            if (control instanceof FormControl) {
                control.markAsTouched({ onlySelf: true });
            } else if (control instanceof FormGroup) {
                this.validateAllFormFields(control);
            }
        });
    }
}

```

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 [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Template in Angular Uploader component

You can customize the default appearance of the uploader using a template along with buttons.

File list template

The `template` property is used to customize the default appearance of each file in the list. It can be represented as the HTML element or string. The selected or dropped files are displayed as per the template layout provided. The remove and progress bar action is handled using the corresponding events when the template is defined.

For example, you can display file type icon along with default UI elements.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component, ViewChild } from '@angular/core';
import { detach } from '@syncfusion/ej2-base';
import { UploaderComponent, FileInfo, SelectedEventArgs } from
 '@syncfusion/ej2-angular-inputs';
/**
 * Uploader Custom Template sample
 */
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id='dropArea'><span id='drop' class="droparea"> Drop
files here or <a href="" id='browse'><u>Browse</u></a> </span>
<ejs-uploader #defaultupload autoUpload='false' [asyncSettings]='path'
(selected)="onSelect($event)" (failure)="onuploadFailed($event)"
(progress)="onFileUpload($event)" (success)="onuploadSuccess($event)">
<ng-template #template let-data="">
  <span class='wrapper'><span class='icon sf-icon-
{{data.type}}'></span>
  <span class='name file-name'>{{data.name}}</span></span>
  <span class='file-size-td file-size'>{{data.size}} bytes</span>
  <span class='e-icons e-file-remove-btn' title='Remove'></span>
<br/>
  <progress id='progressBar' class='progressbar' value='0'
max='100'></progress>
  <span class='percent-td percent'></span>
</ng-template>
</ejs-uploader>
</div> `,
  // styleUrls: ['./index.css']
})
export class AppComponent {
  @ViewChild('defaultupload')
  public uploadObj?: UploaderComponent;
}
```



```

    public path: Object = {
        saveUrl:
            'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
        removeUrl:
            'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
    };

    constructor() {
    }

    ngAfterViewInit() {
        (document.getElementById('browse') as HTMLElement).onclick = function()
        {
            (document.getElementsByClassName('e-file-select-wrap')[0].querySelector('button') as HTMLButtonElement).click();
            return false;
        }
        document.getElementById('dropArea')!.onclick = (e: any) => {
            let target: HTMLElement = <HTMLElement>e.target;
            if (target.classList.contains('e-file-delete-btn')) {
                for (let i: number = 0; i < (this.uploadObj as UploaderComponent).getFilesData().length; i++) {
                    if ((target.closest('li') as HTMLLIElement).getAttribute('data-file-name') === (this.uploadObj as UploaderComponent).getFilesData()[i].name) {
                        (this.uploadObj as UploaderComponent).remove((this.uploadObj as UploaderComponent).getFilesData()[i]);
                    }
                }
            }
            else if (target.classList.contains('e-file-remove-btn')) {
                detach(target.closest('li') as HTMLLIElement);
            }
        }
    }

    public parentElement?: HTMLElement;
    public progressbarContainer?: HTMLElement;
    public filesDetails: FileInfo[] = [];
    public filesList: HTMLElement[] = [];
    public dropElement: HTMLElement =
        document.getElementsByClassName('control-fluid')[0] as HTMLElement;
    public onFileUpload(args: any) {
        let li: HTMLElement = (this.uploadObj as any)!.uploadWrapper?.querySelector('[data-file-name="' + args.file.name + '"]');
        let progressValue: number = Math.round((args.e.loaded / args.e.total) * 100);
        li.getElementsByTagName('progress')[0].value = progressValue;
        li.getElementsByClassName('percent')[0].textContent = progressValue.toString() + " %";
    }
    public onuploadSuccess(args: any) {
        if (args.operation === 'remove') {
            let height: string =
                document.getElementById('dropArea')!.style.height;
            height = (parseInt(height) - 40) + 'px';
            document.getElementById('dropArea')!.style.height = height;
        } else {

```

```

        let li: HTMLElement = (this.uploadObj as
any).uploadWrapper.querySelector('[data-file-name="' + args.file.name +
'"]');
        let progressBar: HTMLElement =
li.getElementsByTagName('progress')[0];
        progressBar.classList.add('e-upload-success');
        li.getElementsByClassName('percent')[0].classList.add('e-upload-
success');
        let height: string =
document.getElementById('dropArea')!.style.height;
        document.getElementById('dropArea')!.style.height = parseInt(height)
- 15 + 'px';
    }
}
public onuploadFailed(args: any) {
    let li: HTMLElement = (this.uploadObj as
any).uploadWrapper.querySelector('[data-file-name="' + args.file.name +
'"]');
    let progressBar: HTMLElement = li.getElementsByTagName('progress')[0];
    progressBar.classList.add('e-upload-failed');
    li.getElementsByClassName('percent')[0].classList.add('e-upload-failed');
}
public onSelect(args: SelectedEventArgs) {
    let length: number = args.filesData.length;
    let height: string = document.getElementById('dropArea')!.style.height;
    height = parseInt(height) + (length * 55) + 'px';
    document.getElementById('dropArea')!.style.height = height;
}
}

```

MAIN.TS

```

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

```

Custom template

You can design the own template by preventing the default file list including buttons.

The `showFileList` property is used to display whether the default file list or own file list when you use custom template to upload or remove the files, pass the custom UI argument as true to call `upload/remove` public method as follows:

- `UploaderObj.upload(filesData, true);`
- `UploaderObj.remove(filesData, true);`

Refer to the following code sample.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'

```

```

import { Component, ViewChild } from '@angular/core';
import { EmitType, detach, isNullOrUndefined, createElement, EventHandler }
from '@syncfusion/ej2-base';
import { UploaderComponent, FileInfo, SelectedEventArgs, RemovingEventArgs }
from '@syncfusion/ej2-angular-inputs';
import { createSpinner, showSpinner, hideSpinner } from '@syncfusion/ej2-
popups';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id='dropArea'>
    <span id='drop' class="droparea"> Drop files here or <a
href="" id='browse'><u>Browse</u></a> </span>
    <ejs-uploader #templateupload id='templatefileupload'
[dropArea]="dropArea" [asyncSettings]='path' (progress)='onFileUpload($event)'
(selected)='onFileSelect($event)' (removing)='onFileRemove($event)'
(failure)='onUploadFailed($event)' (success)='onUploadSuccess($event)'></ejs-
uploader>
    </div>`,
  styleUrls: ['./index.css']
})
export class AppComponent {
  @ViewChild('templateupload')
  public uploadObj?: UploaderComponent;
  public path: Object = {
    saveUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  ngAfterViewInit() {
    this.dropArea = document.querySelector('#dropArea') as HTMLElement;
    (document.getElementById('browse') as HTMLElement).onclick = function() {
      (document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button') as HTMLButtonElement).click();
      return false;
    }
  }
  constructor() {
  }
  public dropArea?: HTMLElement;
  public uploadWrapper: HTMLElement = document.getElementsByClassName('e-
upload')[0] as HTMLElement;
  public parentElement?: HTMLElement;
  public progressBarContainer?: HTMLElement;
  public filesDetails : FileInfo[] = [];
  public fileList: HTMLElement[] = [];
  public onFileSelect(args : SelectedEventArgs) : void {
    if (isNullOrUndefined((document.getElementById('dropArea') as
HTMLElement).querySelector('.upload-list-root') as Element )) {
      this.parentElement = createElement('div', { className: 'upload-
list-root' });
      this.parentElement.appendChild(createElement('ul', {className:
'ul-element' }));
    }
  }
}

```

```

        (document.getElementById('dropArea') as
HTMLElement).appendChild(this.parentElement);
    }
    for (let i : number = 0; i < args.filesData.length; i++) {
        this.formSelectedData(args.filesData[i], this); // create the LI
element for each file Data
    }
    this.filesDetails = this.filesDetails.concat(args.filesData);
    this.uploadObj?.upload(args.filesData, true);
    args.cancel = true;
}

    public formSelectedData (selectedFiles : FileInfo, proxy: any) : void {
        let liEle : HTMLElement = createElement('li', { className: 'file-
lists', attrs: {'data-file-name' : selectedFiles.name} });
        liEle.appendChild(createElement('span', {className: 'file-name ',
innerHTML: selectedFiles.name }));
        liEle.appendChild(createElement('span', {className: 'file-size ',
innerHTML: (this.uploadObj as
UploaderComponent).bytesToSize(selectedFiles.size) }));
        if (selectedFiles.status === 'Ready to upload') {
            this.progressBarContainer = createElement('span', {className:
'progress-bar-container'});
            this.progressBarContainer.appendChild(createElement('progress',
{className: 'progress', attrs: {value : '0', max : '100'}}));
            liEle.appendChild(this.progressBarContainer);
        } else { (liEle.querySelector('.file-name') as
Element).classList.add('upload-fails'); }
        let closeIconContainer : HTMLElement = createElement('span',
{className: 'e-icons close-icon-container'});
        EventHandler.add(closeIconContainer, 'click', this.removeFiles,
proxy);
        liEle.appendChild(closeIconContainer);
        (document.querySelector('.ul-element') as
Element).appendChild(liEle);
        this.filesList.push(liEle);
    }

    public onFileUpload(args : any) : void {
        let li: Element = (document.getElementById('dropArea') as
HTMLElement).querySelector('[data-file-name="' + args.file.name + '"') as
Element;
        EventHandler.remove(li.querySelector('.close-icon-container') as
Element, 'click', this.removeFiles);
        let progressValue : number = Math.round((args.e.loaded /
args.e.total) * 100);
        if (!isNaN(progressValue)) {
            li.getElementsByTagName('progress')[0].value = progressValue;
// Updating the progress bar value
        }
    }

    public onUploadSuccess: EmitType<Object> = (args: any) => {
        let spinnerElement: HTMLElement = document.getElementById('dropArea')
as HTMLElement;
        let li: HTMLElement = (document.getElementById('dropArea') as
HTMLElement).querySelector('[data-file-name="' + args.file.name + '"') as
HTMLElement;
        if (args.operation === 'upload') {

```

```

        let progressBar: HTMLElement =
li.getElementsByTagName('progress')[0];
        (li.querySelector('.close-icon-container') as
Element).classList.add('delete-icon');
        detach(li.getElementsByTagName('progress')[0]);
        (li.querySelector('.file-size') as HTMLElement).style.display =
'inline-block';
        (li.querySelector('.file-name') as HTMLElement).style.color =
'green';
        (li.querySelector('.e-icons') as HTMLElement).onclick = () => {
            createSpinner({ target: spinnerElement, width: '25px' });
            showSpinner(spinnerElement);
        };
        (li.querySelector('.close-icon-container') as
HTMLElement).onkeydown = (e: any) => {
            if (e.keyCode === 13) {
                createSpinner({ target: spinnerElement, width: '25px' });
                showSpinner(spinnerElement);
            }
        };
    } else {
        this.filesList.splice(this.filesList.indexOf(li), 1);
        this.filesDetails.splice(this.filesList.indexOf(li), 1);
        detach(li);
        hideSpinner(spinnerElement);
        detach(spinnerElement.querySelector('.e-spinner-pane') as Element
);
    }
    EventHandler.add(li.querySelector('.close-icon-container') as
Element, 'click', this.removeFiles, this);
}
public onFileRemove(args: RemovingEventArgs): void {
    args.postRawFile = false;
}
public onUploadFailed(args : any) : void {
    let li: Element = (document.getElementById('dropArea') as
HTMLElement).querySelector('[data-file-name="' + args.file.name + '"]') as
Element;
    EventHandler.add(li.querySelector('.close-icon-container') as
Element, 'click', this.removeFiles, this);
    (li.querySelector('.file-name ') as Element).classList.add('upload-
fails');
    if (args.operation === 'upload') {
        detach(li.querySelector('.progress-bar-container') as Element);
    }
}
public removeFiles(args : any) : void {
    let status : string =
this.filesDetails[this.filesList.indexOf(args.currentTarget.parentElement)].s
tatus;
    if (status === 'File uploaded successfully') {
        this.uploadObj?.remove(this.filesDetails[this.filesList.indexOf(args.currentT
arget.parentElement)]);
    } else {
        detach(args.currentTarget.parentElement);
    }
}

```

```
(this.uploadObj as UploaderComponent).element.value = '';
}
public generateSpinner(targetElement: HTMLElement): void {
  createSpinner({ target: targetElement, width: '25px' });
  showSpinner(targetElement);
}
}
```

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 [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

See Also

- [Customize progress bar](#)
- [Customize button with HTML element](#)
- [Customize drop area](#)

Localization in Angular Uploader component

The Localization library allows you to localize static text content of the uploader.

The static text contains default text content of action buttons, file status, clear icon title, tooltips, and text content of drag area. Define the locale object for a culture and assign it to L10n load method.

The following are the list of keys and its values used in the uploader.

Key	Description
----- -----	
Browse	To customize the browse button text.
Clear	To customize the clear button text.
Upload	To customize the upload button text.
dropFilesHint	To customize the drop area text.
uploadFailedMessage	To customize the status text when the file is failed to upload.
uploadSuccessMessage	To customize the status text when the file is uploaded successfully.
removedSuccessMessage	To customize the status text when the file is removed the successfully from the server.
removedFailedMessage	To customize the status text while the file is failed to remove.
inProgress	To customize the status text while the upload is in progress.
readyToUploadMessage	To customize the status text when the file is selected and ready to upload.

- | pauseUpload | To customize the status text while the uploading is paused. |
- | fileUploadCancel | To customize the status text when uploading is cancelled. |
- | invalidMaxFileSize | To customize the status text when the file size is greater than the maximum file size. |
- | invalidFileType | To customize the status text when the file type is invalid. |
- | invalidMinFileSize | To customize the status text when the file size is less than the minimum file size. |
- | remove | To customize tooltip text for remove icon. |
- | cancel | To customize tooltip text for cancel icon. |
- | delete | To customize tooltip text for delete icon. |
- | totalFiles | To customize tooltip text for total files. |
- | size | To customize tooltip text for size. |

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
import { detach, L10n } from '@syncfusion/ej2-base';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-uploader #defaultupload [asyncSettings]='path' locale=
    'fr-CH' autoUpload = 'false'></ejs-uploader>
  `
})
export class AppComponent {
  public path: Object = {
    saveUrl:
    'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
    'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  ngOnInit() {
    L10n.load({
      "fr-CH": {
        "uploader": {
          "invalidMinFileSize" : "La taille du fichier est trop petite! S'il vous plaît télécharger des fichiers avec une taille minimale de 10 Ko",
          "invalidMaxFileSize" : "La taille du fichier dépasse 28 Mo",
          "invalidFileType" : "Le type de fichier n'est pas autorisé",
          "Browse" : "Feuilleter",
          "Clear" : "Clair",
          "Upload" : "Télécharger",
          "dropFilesHint" : "ou Déposer des fichiers ici",
          "uploadFailedMessage" : "Impossible d'importer le fichier",
          "uploadSuccessMessage" : "Fichier téléchargé avec succès",
```

```

        "removedSuccessMessage": "Fichier supprimé avec succès",
        "removedFailedMessage": "Le fichier n'a pas pu être supprimé",
        "inProgress": "Téléchargement",
        "readyToUploadMessage": "Prêt à télécharger",
        "remove": "Retirer",
        "cancel": "Annuler",
        "delete": "Supprimer le fichier",
        "totalFiles": "Total des fichiers",
        "size": "taille"
    }
}
}))
}
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 also explore [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Accessibility in Angular Uploader component

The Uploader 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 Uploader 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>

```

Keyboard interaction

The following are the standard keys that works on uploader component.

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| **Tab** | Move focus to next element. |

| **Shift + Tab** | Move focus to previous element. |

| **Enter** | Triggers corresponding action to button element. |

| **Esc** | Close the file browser dialog alone and cancels the upload on drop the file. |

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-uploader #defaultupload [asyncSettings]='path'></ejs-
uploader>

```

```
  })
  export class AppComponent {
    public path: Object = {
      saveUrl:
        'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
      removeUrl:
        'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
    };
    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 also explore [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion Angular components](#)

Style appearance in Angular Uploader 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 File Upload wrapper element

Use the following CSS to customize the appearance of wrapper element.

```
`css
/ To specify height /
.e-upload.e-control-wrapper, .e-bigger.e-small .e-upload.e-control-wrapper {
height: 300px;
width: 300px;
}
```

`

Customizing the File Upload browse button

Use the following CSS to customize the File Upload browse button

```
`css
```

```
/ To specify font size and color /
```

```
.e-upload .e-file-select-wrap .e-btn, .e-upload .e-upload-actions .e-btn, .e-bigger.e-small .e-upload .e-file-select-wrap .e-btn, .e-bigger.e-small .e-upload .e-upload-actions .e-btn {
```

```
font-family: cursive;
```

```
height: 40px;
```

```
background-color: aquamarine;
```

```
color: coral;
```

```
}
```

`

Customizing the File Upload content

Use the following CSS to customize the File Upload content

```
`css
```

```
/ To specify font size and color /
```

```
.e-upload .e-file-select-wrap .e-file-drop, .e-bigger.e-small .e-upload .e-file-select-wrap .e-file-drop {
```

```
font-size: 20px;
```

```
color: aqua;
```

```
}
```

`

Customizing the uploaded file container in File Upload

Use the following CSS to customize the uploaded file container in File Upload

```
`css
```

```
/ To specify background color /
```

```
.e-upload .e-upload-files .e-upload-file-list {
```

```
background-color: beige;
```

```
}
```

`

See Also

- [Customize the appearance of uploader using a template](#)

How To

Hide default drop area in Angular Uploader component

You can achieve this behavior by override the corresponding uploader styles. In the following example, override the below styles to hide the default drop area behavior.

- .e-upload.e-control
- .e-upload .e-file-select
- .e-upload .e-file-drop

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: 'default.html',
  styleUrls: ['./index.css']
})
export class AppComponent {
  public path: Object = {
    saveUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
}
```

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 [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Preview images before uploading in Angular Uploader component

The uploader component allows to create preview images before uploaded it. The preview images created by reading the file using selected event. Also, the user can create preview images after uploading to server using success event. Refer to the following link to learn about how to create image preview.

[Image Preview](#)

You can also explore [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Achieve invisible upload in Angular Uploader component

You can achieve the invisible upload feature by using selected event in uploader component.

Refer to the following example.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
import { createElement } from '@syncfusion/ej2-base';
import { SelectedEventArgs } from '@syncfusion/ej2-angular-inputs';
/**
 * Default Uploader Default Component
 */
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './default.html',
  styleUrls: ['./index.css']
})
export class AppComponent {
  public locale: string = 'en-US';
  public path: Object = {
    saveUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  public allowExtensions: string = '.png, .jpg, .jpeg';
  public onSelected(args: SelectedEventArgs):void {
    for(let i = 0; i< args.filesData.length ; i++){
      let liparentDiv = createElement('div', { className: 'image-list'});
      let liImage = createElement('img', { className: 'image'});
      liparentDiv.appendChild(liImage);
      this.readURL(liImage, args.filesData[i]);
      (document.getElementById('preview') as
      HTMLElement).appendChild(liparentDiv);
    }
    args.cancel=true;
  }
  public readURL(liImage: HTMLElement, file: any):void {
    let imgPreview: HTMLImageElement | any = liImage as HTMLImageElement;
    let imageFile: File = file.rawFile;
    let reader: FileReader = new FileReader();
    reader.addEventListener( 'load', () => {
      imgPreview.src = reader.result;
    }, false);
    if (imageFile) {
```

```
        reader.readAsDataURL(imageFile);  
    }  
};  
}
```

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 [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Customize progressbar in Angular Uploader component

You can customize the progress bar's size, color, and background by overriding the styles in uploader component. Refer to the following example.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core';  
import { BrowserModule } from '@angular/platform-browser';  
import { UploaderModule } from '@syncfusion/ej2-angular-inputs';  
import { Component } from '@angular/core';  
@Component({  
  imports: [  
    UploaderModule  
  ],  
  standalone: true,  
  selector: 'app-root',  
  templateUrl: './default.html',  
  styleUrls: ['./index.css']  
})  
export class AppComponent {  
  public path: Object = {  
    saveUrl:  
      'https://services.syncfusion.com/angular/production/api/FileUploader/Save',  
    removeUrl:  
      'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'  
  };  
}
```

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 [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Sort the selected files in Angular Uploader component

You can sort the selected files in an uploader component by using the [selected](#) event. Refer to the following example.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component, ViewChild } from '@angular/core';
import { UploaderComponent, SelectedEventArgs, FileInfo } from
 '@syncfusion/ej2-angular-inputs';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: 'default.html',
  styleUrls: ['index.css']
})
export class AppComponent {
  @ViewChild('defaultupload')
  public uploadObj?: UploaderComponent;
  public path: Object = {
    saveUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  public initial:boolean = true;
  public onSelect(args: SelectedEventArgs): void {
    if (this.initial) { this.initial = false; return; }
    args.isModified = true;
    let oldFiles: FileInfo[] = this.uploadObj?.getFilesData() as FileInfo[];
    let filesData: FileInfo[] = args.filesData.concat(oldFiles);
    let modifiedData: FileInfo[] = this.sortFileList(filesData);
    args.modifiedFilesData = modifiedData;
  }
  public sortFileList(filesData: FileInfo[]): FileInfo[] {
    let files: FileInfo[] = filesData;
    let fileNames: string[] = [];
    for (let i: number = 0; i < files.length; i++) {
      fileNames.push(files[i].name);
    }
    let sortedFileNames: string[] = fileNames.sort();
    let sortedFilesData: FileInfo[] = [];
    let index: number = 0;
    for (let name of sortedFileNames) {
      for (let i: number = 0; i < files.length; i++) {
        if (name === files[i].name) {
          sortedFilesData.push(files[i]);
        }
      }
    }
  }
}
```

```

    }
  }
  return sortedFilesData;
}
}

```

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 [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Get the total size of selected files in Angular Uploader component

You can get the total size of selected files before uploading it to the designated server.

This can be achieved by using the selected event. Refer to the following example to calculate the total file size.

In the following example, explains about how to calculate total file size before upload.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component, ViewChild } from '@angular/core';
import { UploaderComponent, SelectedEventArgs } from '@syncfusion/ej2-angular-inputs';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './default.html',
  styleUrls: ['./index.css']
})
export class AppComponent {
  @ViewChild('defaultupload')
  public uploadObj?: UploaderComponent;
  public path: Object = {
    saveUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  public onSelect(args: SelectedEventArgs): void {
    let totalSize: number = 0;
    for (let file of args.filesData) {
      totalSize = totalSize + file.size;
    }
  }
}

```



```
let size: string = (this.uploadObj as
UploaderComponent).bytesToSize(totalSize);
alert("Total select file's size is " + size)
}
}
```

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 [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Customize button with html element in Angular Uploader component

The uploader component allows you to customize the action buttons by using the [buttons](#) property. Refer to the following example.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
import { createElement } from '@syncfusion/ej2-base';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './default.html',
  styleUrls: ['./index.css']
})
export class AppComponent {
  public path: Object = {
    saveUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  public uploadEle: HTMLElement = createElement('span', { className: 'upload
e-icons', innerHTML: 'Upload All' });
  public clearEle = createElement('span', { className: 'remove e-icons',
innerHTML: 'Clear All' });
  public buttons: Object = {
    browse: 'Choose file',
    clear: this.clearEle,
    upload: this.uploadEle
  };
}
```

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 [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Add confirm dialog to remove the files in Angular Uploader component

You can customize the uploader component using confirm dialog before removing the files.

Here, ej2 dialog is used as confirm dialog. Refer to the following example.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { Component, ViewChild } from '@angular/core';
import { EmitType } from '@syncfusion/ej2-base';
import { UploaderComponent, SelectedEventArgs } from '@syncfusion/ej2-angular-inputs';
import { DialogComponent } from '@syncfusion/ej2-angular-popups';
@Component({
  imports: [ UploaderModule, DialogModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './default.html',
  styleUrls: ['./index.css']
})
export class AppComponent {
  @ViewChild('defaultupload') uploadObj?: UploaderComponent;
  @ViewChild('dialog') dialog?: DialogComponent;
  public path: Object = {
    saveUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  public removeFile: any = [];
  public content: string = 'Confirm to remove the file?';
  public width: string = '250px';
  public visible: boolean = false;
  public target: string = '#container';
  public buttons: Object = [{ 'click': this.onClick.bind(this), buttonModel:
    { content: 'OK', cssClass: 'e-flat', isPrimary: true }},
    { 'click': () => {this.dialog?.hide(); }, buttonModel: { content: 'Cancel',
    cssClass: 'e-flat' } }];
  public onremoving: EmitType<SelectedEventArgs> = (args: any) => {
    args.cancel = true;
    this.removeFile.push(args.filesData);
    this.dialog?.show();
  }
```

```
};
onClick() {
  this.dialog?.hide();
  this.uploadObj?.remove(this.removeFile, true);
  this.removeFile = [];
}
}
```

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 [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Add additional data on upload in Angular Uploader component

The uploader component allows you to add additional data on file upload, which is used to get in the server-side.

By using the [uploading](#) event and its customFormData argument, you can achieve this behavior. Refer to the following example

In the following code snippet, explains about how to add additional data on file upload.

```
`typescript
import { Component } from '@angular/core';
import { EmitType } from '@syncfusion/ej2-base';
import { SelectedEventArgs } from '@syncfusion/ej2-angular-inputs';
@Component({
  selector: 'app-root',
  templateUrl: 'default.html',
  styleUrls: ['index.css']
})
export class AppComponent {
  public path: Object = {
    saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save',
    removeUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Remove'
  };
  public onFileUpload: EmitType<SelectedEventArgs> = (args: any) => {
    // add addition data as key-value pair.
```

```
args.customFormData = [{ 'name': 'Syncfusion INC' }];
};
}
`
```

Server side for adding additional data

```
`csharp
// Get the additional data in server end by corresponding key.
var data = HttpContext.Current.Request.Form["name"];
`
```

You can also explore [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Validate image on drop in Angular Uploader component

The uploader component allows you to upload all type of images by setting **image/*** to [allowedExtensions](#) property.

You can directly set it to **accept** attribute of uploader element.

By default, the behavior is working with select a file using browse button. But, this behavior doesn't support on drag and drop the files. You can handle this behavior manually using **selected** event by filtering the file types from application.

In the following example, validated image files using images/*. You are able to drag and drop the image files with extension of PNG, JPG, BPG, GIF and TIFF to upload it.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
import { EmitType } from '@syncfusion/ej2-base';
import { SelectedEventArgs, UploaderComponent } from '@syncfusion/ej2-angular-inputs';
import { } from '@syncfusion/ej2-angular-inputs';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './default.html',
  styleUrls: ['./index.css']
})
export class AppComponent {
  public path: Object = {
    saveUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
```

```

    };
    public onSelected: EmitType<SelectedEventArgs> = (args: any) => {
        if (event!.type === 'drop') {
            let allImages: Array<string> = ['png', 'jpg', 'jpeg', 'gif',
            'tiff', 'bpg'];
            let files = args.filesData;
            let modifiedFiles = [];
            for (let file of files) {
                if (allImages.indexOf(file.type) === -1) {
                    file.status = 'File type is not allowed';
                    file.statusCode = '0';
                }
                modifiedFiles.push(file);
            }
            args.isModified = 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 also explore [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Determine whether uploader has file input in Angular Uploader component

By setting **required** attribute to uploader input element, you can validate the file input has any value in it.

In the below sample, set required attribute to the uploader input element and showcase the validation failure message using `data-required-message` attribute.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { Component, ViewChild, AfterViewInit } from '@angular/core';
import { EmitType, createElement, detach } from '@syncfusion/ej2-base';
import { UploaderComponent, SelectedEventArgs, FormValidator,
FormValidatorModel } from '@syncfusion/ej2-angular-inputs';
import { DialogComponent } from '@syncfusion/ej2-angular-popups';
@Component({
  imports: [
    FormsModule, UploaderModule, DialogModule
  ],
  standalone: true,
  selector: 'app-root',

```

```

        templateUrl: './default.html',
        styleUrls: ['./index.css']
    })
    export class AppComponent {
        @ViewChild('defaultupload')
        public uploadObj?: UploaderComponent;
        @ViewChild('dialog')
        public dialog?: DialogComponent;
        public path: Object = {
            saveUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
            removeUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
        };
        public removeFile: any = [];
        public content: string = 'Your details have been updated successfully,
Thank you.';
        public width: string = '250px';
        public visible: boolean = false;
        public header: string = 'Success';
        public target: HTMLElement = document.getElementById('control_wrapper')
    as HTMLElement;
        public onFileSelect: EmitType<SelectedEventArgs> = (args: any) => {
            if (args.filesData.length > 0) {
                if (document.getElementsByClassName('upload-image').length > 0) {
                    detach(document.getElementsByClassName('imgWrapper')[0]);
                }
                let imageTag = createElement('IMG', { className: 'upload-image',
attrs: { 'alt': 'Image' } });
                let wrapper: HTMLElement = createElement('span', { className:
'imgWrapper' }) as HTMLElement;
                wrapper.appendChild(imageTag);
                let rootFile = document.getElementsByClassName('dropUpload')[0];
                rootFile.insertBefore(wrapper, rootFile.firstChild);
                this.readURL(wrapper, args.filesData[0]);
            }
            args.cancel = true;
        }
        options: FormValidatorModel | undefined;
        autoUpload: any;
        extensions: any;
        dropElement: any;
        ngAfterViewInit() {
            (document.getElementById('customBrowse') as HTMLElement).onclick = ()
=> {
                (document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button') as HTMLButtonElement).click();
            };
            (document.getElementById('submit-btn') as HTMLElement).onclick = ()
=> {
                this.onFormSubmit();
            };
            let options = { rules: { 'name': { required: true } } };
            let formObj: FormValidator = new FormValidator('#form1',
this.options);
            let proxy = this;
            setTimeout(() => {

```

```

        this.uploadObj?.element.setAttribute('data-required-message', '*
Choose your image to upload');
        this.uploadObj?.element.setAttribute('required', '');
        this.uploadObj?.element.setAttribute('data-msg-containerid',
'uploadError');
    }, 500);
    }
    public readURL: EmitType<SelectedEventArgs> = (li: HTMLElement, args:
any) => {
        let preview: HTMLImageElement = li.querySelector('.upload-image') as
HTMLImageElement;
        let file: File = args.rawFile; let reader: FileReader = new
FileReader();
        reader.addEventListener('load', () => { preview.src = reader.result
as string; }, false);
        if (file) { reader.readAsDataURL(file); }
    }
    public onFormSubmit: any = () => {
        let formObj: FormValidator = new FormValidator('#form1',
this.options);
        let formStatus: Boolean = formObj.validate();
        if (formStatus) {
            formObj.element.reset();
            detach(document.getElementsByClassName('imgWrapper')[0]);
            this.dialog?.show();
        }
    }
}

```

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 [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Achieve file upload programmatically in Angular Uploader component

You can upload a file programmatically using [upload](#) method.

The selected files data, get from [getFilesData](#) public method in uploader.

The upload method behaves differently based on its arguments.

- If this method receives any files as arguments, those files only start to upload.
- If it has no argument then all the selected files are will start to upload.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'

```

```
import { Component, ViewChild, AfterViewInit } from '@angular/core';
import { UploaderComponent } from '@syncfusion/ej2-angular-inputs';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './default.html',
  styleUrls: ['./index.css']
})
export class AppComponent {
  @ViewChild('defaultupload')
  public uploadObj?: UploaderComponent;
  public autoUpload: boolean = false;
  public path: Object = {
    saveUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  ngAfterViewInit(): void {
    (document.getElementById('first') as HTMLElement).onclick = (args) =>
    {
      (this.uploadObj as UploaderComponent).upload((this.uploadObj as
UploaderComponent).getFilesData()[0]);
    };
    (document.getElementById('full') as HTMLElement).onclick = (args) =>
    {
      this.uploadObj?.upload((this.uploadObj as
UploaderComponent).getFilesData());
    };
  }
}
```

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 [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Check file size before uploading it in Angular Uploader component

By using [uploading](#) event, you can get the file size before upload it to server.

File object contains the file size in bytes only.

You can convert the size to standard formats (KB or MB) using [bytesToSize](#) method.

APP.COMPONENT.TS

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



```
import { EmitType } from '@syncfusion/ej2-base';
import { UploaderComponent } from '@syncfusion/ej2-angular-inputs';
@Component({
  selector: 'app-root',
  templateUrl: './default.html',
  styleUrls: ['./index.css']
})
export class AppComponent {
  @ViewChild('defaultupload')
  public uploadObj?: UploaderComponent;
  public autoUpload: boolean = false;
  public path: Object = {
    saveUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  public onBeforeUpload: EmitType<Object> = (args: any) => {
    // get the file size in bytes
    let sizeInBytes: number = args.fileData.size;
    // get the file size in standard format
    alert("File size is: " + this.uploadObj?.bytesToSize(sizeInBytes));
  }
}
```

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 [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Check the mime type of file before upload it in Angular Uploader component

By using [uploading](#) event, you can get the file MIME type before uploading it to server.

In the below sample, file MIME type is shown in the alert box before file start to upload.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
import { EmitType } from '@syncfusion/ej2-base';
@Component({
  imports: [
    UploaderModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './default.html',
  styleUrls: ['./index.css']
})
```

```

    })
    export class AppComponent {
        public autoUpload: boolean = false;
        public path: Object = {
            saveUrl:
                'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
            removeUrl:
                'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
        };
        public onBeforeUpload: EmitType<Object> = (args: any) => {
            // get the file MIME type
            alert("File MIME type is: " + args.fileData.rawFile.type)
        }
    }
}

```

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 [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Trigger click event of input file in Angular Uploader component

You can trigger the click event of input file from external button using `click` event of button. In the below sample, triggered click event of input file from `Essential JavaScript 2 Button`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component, AfterViewInit } from '@angular/core';
@Component({
    imports: [
        UploaderModule
    ],
    standalone: true,
    selector: 'app-root',
    templateUrl: './default.html',
    styleUrls: ['./index.css']
})
export class AppComponent {
    public autoUpload: boolean = false;
    public path: Object = {
        saveUrl:
            'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
        removeUrl:
            'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
    };
    ngAfterViewInit(): void {

```

```
(document.getElementById('browse') as HTMLElement).onclick = (args)
=> {
    (document.getElementsByClassName('e-file-select-wrap')[0].querySelector('button') as HTMLButtonElement).click();
};
}
```

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 [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Open and edit the uploaded files in Angular Uploader component

The uploader component allows you to modify the file after uploading to the server, which can be achieved using success event of the uploader.

You can retrieve the saved file path in the uploader success event and assign it to custom attribute (data-file-name) value of the respective file list element to open the uploaded file. Click the respective file element to create a new request along with saved file path using http header. In the server-side, get the file path from the header and open the file using `process.start` method.

`typescript

```
import { Component } from '@angular/core';
import { EmitType } from '@syncfusion/ej2-base';

@Component({
  selector: 'app-root',
  template: '<div class="control_wrapper"> <ejs-uploader #defaultupload id="fileupload" [asyncSettings]="path" (success)="onUploadSuccess($event)"></ejs-uploader></div>'
})
export class AppComponent {
  public path: Object = {
    saveUrl: 'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl: 'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  public onUploadSuccess: EmitType<Object> = (args: any) => {
    let liElements: any = document.body.querySelectorAll('.e-upload-file-list');
    for (let i = 0; i < liElements.length; i++) {
```

```

if (liElements[i].getAttribute('data-file-name') == args.file.name) {
liElements[i].addEventListener('click', () => { this.openFile(args, event); });
// File path have to update from server end in response status description.
liElements[i].setAttribute('file-path', args.e.target.statusText);
}
}
};
openFile(args: any, e: any) {
if (!e.target.classList.contains('e-file-delete-btn') && !e.target.classList.contains('e-file-remove-btn'))
{
let ajax = new XMLHttpRequest();
// create new request for open the selected file
ajax.open("POST", '/Home/openFile');
let liElements = document.getElementsByClassName('e-upload')[0].querySelectorAll('.e-upload-file-list');
for (let i = 0; i < liElements.length; i++) {
if (liElements[i].getAttribute('data-file-name') == args.file.name) {
// Added the file path in header to get it in server side.
ajax.setRequestHeader('filePath', liElements[i].getAttribute('file-path').toString());
}
}
ajax.send();
}
}
}
`

```

Server side for open and edit the uploaded files

`csharp

```

public void Save() {
if (!System.IO.File.Exists(fileSavePath))
{
httpPostedFile.SaveAs(fileSavePath);
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();

```

```

Response.ContentType = "application/json; charset=utf-8";
// Sending the file path to client side
Response.StatusDescription = fileSavePath;
Response.End();
}
}
[AcceptVerbs("Post")]
public void openFile()
{
// Check whether the file is available in the corresponding location
if (System.IO.File.Exists(Request.Headers.GetValues("filePath").First()))
{
// This will open the selected file from server location in desktop
Process.Start(Request.Headers.GetValues("filePath").First());
}
}
`

```

You can also explore [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

[Resize images before upload to server.](#)

You can customize the dimension of the images before uploading it to the server.

By using selected event, you can get the selected file information as type of an object. From the obtained image file information, create a new canvas and render an image with the custom dimensions. Refer the corresponding code snippet as follows.

```

`typescript
@ViewChild('templateupload')
public uploadObj: UploaderComponent;
public path: Object = {
saveUrl: 'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
removeUrl: 'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
};
public uploadWrapper: HTMLElement = document.getElementsByClassName('e-upload')[0] as
HTMLElement;
public parentElement : HTMLElement;

```

```
public proxy : any;
public progressBarContainer : HTMLElement;
public filesDetails : FileInfo[] = [];
public fileList: HTMLElement[] = [];
public dropElement: HTMLElement = document.getElementsByClassName('control-fluid')[0] as
HTMLElement;
public newImage: any;
ngAfterViewInit(): void {
document.getElementById('browse').onclick = () => {
document.getElementsByClassName('e-file-select-wrap')[0].querySelector('button').click();
return false;
};
}
public onFileSelect(args : SelectedEventArgs) : void {
args.cancel = true;
if (isNullOrUndefined(document.getElementById('dropArea').querySelector('.upload-list-root'))) {
this.parentElement = createElement('div', { className: 'upload-list-root' });
this.parentElement.appendChild(createElement('ul', {className: 'ul-element' }));
document.getElementById('dropArea').appendChild(this.parentElement);
}
for (let i : number = 0; i < args.filesData.length; i++) {
this.formSelectedData(args.filesData[i], this); // create the LI element for each file Data
}
this.filesDetails = this.filesDetails.concat(args.filesData);
let proxy = this;
let file: FileInfo = args.filesData[0].rawFile as any;
let width: number;
let height: number;
let img: any = document.createElement("img");
let reader: any = new FileReader();
reader.onload = function(e: any) { img.src = e.target.result; };
reader.readAsDataURL(file);
let imgs = new Image();
```

```
img.onload = function(): void {
width = this.width;
height = this.height;
proxy.onNewImg(height, width, img, args.filesData[0])
};
imgs.src = img.src;
}
// to create canvas and update our custom dimensions
private onNewImg(height: any, width: any, img: any, file: any) {
let canvas: HTMLCanvasElement = document.createElement("canvas");
let ctx: any = canvas.getContext("2d");
ctx.drawImage(img, 0, 0);
let MAX_WIDTH: any = 1000;
let MAX_HEIGHT: any = 600;
if (width > height) {
if (width > MAX_WIDTH) {
height *= MAX_WIDTH / width;
width = MAX_WIDTH;
}
} else {
if (height > MAX_HEIGHT) {
width *= MAX_HEIGHT / height;
height = MAX_HEIGHT;
}
}
canvas.width = width;
canvas.height = height;
let ctx1 = canvas.getContext("2d");
ctx1.drawImage(img, 0, 0, width, height);
this.newImage = canvas.toDataURL("image/png");
let blobBin = atob(this.newImage.split(',')[1]);
let array = [];
for(var i = 0; i < blobBin.length; i++) {
```

```
array.push(blobBin.charCodeAt(i));
}
let newBlob = new Blob([new Uint8Array(array)], {type: 'image/png'});
let newFile: any = this.createFile(newBlob, file);
this.uploadObj.upload(newFile, true);
}
// To create File object to upload
public createFile(image: any , file: any) {
let newFile = {
name: file.name,
rawFile: image,
size: image.size,
type: file.type,
validationMessage: '',
statusCode: '1',
status: 'Ready to Upload'
}
return newFile;
}
public onFileRemove(args: RemovingEventArgs): void {
args.postRawFile = false;
}
public formSelectedData (selectedFiles : FileInfo, proxy: any ) : void {
let liEle : HTMLElement = createElement('li', { className: 'file-lists', attrs: {'data-file-name' :
selectedFiles.name} });
liEle.appendChild(createElement('span', {className: 'file-name ', innerHTML: selectedFiles.name }));
liEle.appendChild(createElement('span', {className: 'file-size ', innerHTML:
this.uploadObj.bytesToSize(selectedFiles.size) }));
if (selectedFiles.status === 'Ready to upload') {
this.progressbarContainer = createElement('span', {className: 'progress-bar-container'});
this.progressbarContainer.appendChild(createElement('progress', {className: 'progress', attrs: {value :
'0', max : '100'}} ));
liEle.appendChild(this.progressbarContainer);
} else { liEle.querySelector('.file-name').classList.add('upload-fails'); }
```



```
let closeIconContainer : HTMLElement = createElement('span', {className: 'e-icons close-icon-
container'}));

EventHandler.add(closeIconContainer, 'click', this.removeFiles, proxy);

liEle.appendChild(closeIconContainer);

document.querySelector('.ul-element').appendChild(liEle);

this.filesList.push(liEle);
}

public onFileUpload(args : any) : void {

let li : Element = document.getElementById('dropArea').querySelector('[data-file-name="" +
args.file.name + "']");

EventHandler.remove(li.querySelector('.close-icon-container'), 'click', this.removeFiles);

let progressValue : number = Math.round((args.e.loaded / args.e.total) * 100);

if (!isNaN(progressValue)) {

li.getElementsByTagName('progress')[0].value = progressValue; // Updating the progress bar value
}

}

public onUploadSuccess: EmitType<Object> = (args: any) => {

console.log("The selected file has resized and uploaded");

let spinnerElement: HTMLElement = document.getElementById('dropArea');

let li: Element = document.getElementById('dropArea').querySelector('[data-file-name="" +
args.file.name + "']");

if (args.operation === 'upload') {

let progressBar: HTMLElement = li.getElementsByTagName('progress')[0];

li.querySelector('.close-icon-container').classList.add('delete-icon');

detach(li.getElementsByTagName('progress')[0]);

(li.querySelector('.file-size') as HTMLElement).style.display = 'inline-block';

(li.querySelector('.file-name') as HTMLElement).style.color = 'green';

(li.querySelector('.e-icons') as HTMLElement).onclick = () => {

createSpinner({ target: spinnerElement, width: '25px' });

showSpinner(spinnerElement);

};

(li.querySelector('.close-icon-container') as HTMLElement).onkeydown = (e: any) => {

if (e.keyCode === 13) {

createSpinner({ target: spinnerElement, width: '25px' });
```

```
showSpinner(spinnerElement);
}
};
} else {
this.filesList.splice(this.filesList.indexOf(li), 1);
this.filesDetails.splice(this.filesList.indexOf(li), 1);
if (!NullOrUndefined(li)) { detach(li); }
if (!NullOrUndefined(spinnerElement)) {
hideSpinner(spinnerElement);
detach(spinnerElement.querySelector('.e-spinner-pane'));
}
}
EventHandler.add(li.querySelector('.close-icon-container'), 'click', this.removeFiles, this);
}
public onUploadFailed(args : any) : void {
let li : Element = document.getElementById('dropArea').querySelector('[data-file-name="" +
args.file.name + "']');
EventHandler.add(li.querySelector('.close-icon-container'), 'click', this.removeFiles, this);
li.querySelector('.file-name ').classList.add('upload-fails');
if (args.operation === 'upload') {
detach(li.querySelector('.progress-bar-container'));
}
}
public removeFiles(args : any) : void {
let status : string = this.filesDetails[this.filesList.indexOf(args.currentTarget.parentElement)].status;
if (status === 'File uploaded successfully') {
this.uploadObj.remove(this.filesDetails[this.filesList.indexOf(args.currentTarget.parentElement)]);
} else {
detach(args.currentTarget.parentElement);
}
this.uploadObj.element.value = "";
}
public generateSpinner(targetElement: HTMLElement): void {
```

```

createSpinner({ target: targetElement, width: '25px' });
showSpinner(targetElement);
}
,
,

<div class="uploadtemplate">
<div id='dropArea'>
<span id='drop' class="droparea"> Drop files here or <a href="" id='browse'><u>Browse</u></a>
</span>

<ejs-uploader #templateupload id='templatefileupload' [asyncSettings]='path'
[dropArea]='dropElement' (progress)='onFileUpload($event)' (selected)='onFileSelect($event)'
(failure)='onUploadFailed($event)' (success)='onUploadSuccess($event)'
(removing)='onFileRemove($event)'></ejs-uploader>

</div>
</div>
,
,

<style>
.uploadtemplate #dropArea {
min-height: 50px;
margin: 15px 0;
position: relative;
}
.uploadtemplate #drop {
padding: 3% 30% 3%;
display: inherit;
border: 1px dashed #c3c3cc
}
.e-upload {
float: none;
}
.uploadtemplate .droparea {
font-size: 14px;
}

```

```
.uploadtemplate .e-file-select-wrap {  
display: none;  
}  
.uploadtemplate .e-upload {  
float: none;  
border: none;  
}  
.uploadtemplate .ul-element {  
list-style: none;  
width: 100%;  
padding-left: 0;  
}  
.uploadtemplate .file-name {  
padding: 8px 6px 8px 0;  
font-size: 13px;  
width: 46%;  
display: inline-block;  
position: relative;  
top: 4px;  
}  
.uploadtemplate .file-size {  
padding: 4px;  
font-size: 13px;  
width: 18%;  
display: inline-block;  
position: relative;  
}  
.uploadtemplate li.file-lists {  
border: 1px solid lightgray;  
padding: 0 6px 0 14px;  
margin-top: 15px;  
position: relative;  
background: rgba(0, 0, 0, 0.04);
```

```
}  
.uploadtemplate span.file-size, .file-name {  
font-family: "Helvetica Neue", "Helvetica", "Arial", "sans-serif";  
text-overflow: ellipsis;  
overflow: hidden;  
white-space: nowrap;  
}  
.uploadtemplate span.progress-bar-container {  
display: block;  
float: right;  
height: 20px;  
right: 13%;  
top: 14px;  
position: relative;  
width: 20%;  
}  
.uploadtemplate .progress {  
width: 100%;  
height: 15px;  
-webkit-appearance: none;  
}  
.uploadtemplate .close-icon-container {  
cursor: pointer;  
font-size: 11px;  
height: 24px;  
margin: 0 12px 0 22px;  
padding: 0;  
position: absolute;  
right: 0;  
width: 24px;  
top: 6px;  
}  
.uploadtemplate .close-icon-container.e-icons::before {
```

```
left: 7px;
position: inherit;
top: 7px;
content: '\e932';
}
.uploadtemplate .close-icon-container.delete-icon::before {
content: '\e94a';
}
.uploadtemplate .close-icon-container:hover {
background-color: rgba(0, 0, 0, 0.12);
border-color: transparent;
border-radius: 50%;
box-shadow: 0 0 0 transparent;
}
.uploadtemplate .upload-success {
color: #2bc700;
}
.uploadtemplate .upload-fails {
color: #f44336;
}
.uploadtemplate progress::-webkit-progress-bar {
border: 1px solid lightgrey;
background-color: #ffffff;
border-radius: 2px;
}
.uploadtemplate #dropArea progress {
border: 1px solid lightgrey;
background-color: #ffffff;
border-radius: 2px;
}
.uploadtemplate progress::-webkit-progress-value, .uploadtemplate progress::-webkit-progress-value {
border-radius: 2px;
background-color: #ff4081;
```

```
}  
</style>  
,
```

You can also explore [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Convert image into binary format after uploading in Angular Uploader component

By default, the file upload component saves the uploaded image files in physical directories. Also, you can convert the images into binary format at server-side before saving the uploaded images.

To retrieve binary format of image files, convert the posted file's input stream into binary reader and read as bytes using ReadBytes method.

Refer to the below server-side code snippet

```
`csharp  
[AcceptVerbs("Post")]  
public void Save()  
{  
    try  
    {  
        if (System.Web.HttpContext.Current.Request.Files.AllKeys.Length > 0)  
        {  
            var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];  
            if (httpPostedFile != null)  
            {  
                byte[] fileBytes;  
                using (BinaryReader br = new BinaryReader(httpPostedFile.InputStream))  
                {  
                    fileBytes = br.ReadBytes((int)httpPostedFile.InputStream.Length);  
                    // bytes will be stored in variable fileBytes  
                }  
                HttpResponseMessage Response = System.Web.HttpContext.Current.Response;  
                Response.Clear();  
                Response.ContentType = "application/json; charset=utf-8";  
                Response.StatusCode = 200;  
                Response.Status = "200 Success";  
                Response.End();  
            }  
        }  
    }  
}
```

```

}
}
}
catch (Exception e)
{
    HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
    Response.Clear();
    Response.ContentType = "application/json; charset=utf-8";
    Response.StatusCode = 204;
    Response.Status = "204 No Content";
    Response.StatusDescription = e.Message;
    Response.End();
}
}
`

```

You can also explore [Angular File Upload](#) feature tour page for its groundbreaking features. You can also explore our [Angular File Upload example](#) to understand how to browse the files which you want to upload to the server.

Ej1 api migration in Angular Uploader component

This article describes the API migration process of File Upload component from Essential JS 1 to Essential JS 2.

Accessibility

<!-- markdownlint-disable MD033 -->

Behavior	Property in Essential JS 1	Property in Essential JS 2
Localization	Property : locale <ej-uploadbox id='uploadDefault' locale='es-ES'> </ej-uploadbox>	Property : locale <ejs-uploader locale='es-ES'></ejs-uploader>
Right to left	Property : enableRTL <ej-uploadbox id='uploadDefault' [enableRTL]='true'></ej-uploadbox>	Property : enableRTL <ejs-uploader [enableRtl]='true'> </ejs-uploader>

File List

<!-- markdownlint-disable MD033 -->

Behavior	Property in Essential JS 1	Property in Essential JS 2

| Show/Hide the selected files | **Property** : showFileDetails
 <ej-uploadbox id='uploadDefault' [showFileDetails]= 'false'></ej-uploadbox> | **Property** : showFileList
 <ejs-uploader [showFileList]= 'false'></ejs-uploader> |

| Customizing the file list | Not Applicable | **Property** : template
 <ejs-uploader> <ng-template #template let-data="> // your custom template here</ng-template></ejs-uploader> |

| Get the files in sorted form | Not Applicable | **Method**: SortFileList
 <ejs-uploader #defaultupload></ejs-uploader>
 TS:
 @ViewChild('defaultupload') public uploadObj: UploaderComponent; uploadObj.sortFileList(files) |

| Clearing File List | Not Applicable | **Method**: ClearAll
 <ejs-uploader #defaultupload></ejs-uploader>
 TS:
 @ViewChild('defaultupload') public uploadObj: UploaderComponent;
uploadObj.clearAll() |

| Event triggers when clearing Files | Not Applicable | **Event** : clearing
 <ejs-uploader #defaultupload (clearing)='onClearing(\$event)'></ej-uploadbox>
 TS:
 public onClearing: EmitType< ClearingEventArgs>= () => { }; |

File selection

<!-- markdownlint-disable MD033 -->

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

| ----- | ----- | ----- |

| Select multiple files to upload | **Property** : multipleFilesSelection
 <ej-uploadbox id='uploadDefault' [multipleFilesSelection]= 'true'></ej-uploadbox> | **Property** : multiple
 <ejs-uploader [multiple]='true'></ejs-uploader> |

| Set minimum file size to upload | **Not Applicable** | **Property** : minFileSize
 <ejs-uploader [minFileSize]=1024></ejs-uploader> |

| Set maximum file size to upload | **Property** : fileSize
 <ej-uploadbox id='uploadDefault' [fileSize]= 5000></ej-uploadbox> | **Property** : maxFileSize
 <ejs-uploader [maxFileSize]= 5000></ejs-uploader> |

| Allowed file types to select | **Property** : extensionsAllow
 <ej-uploadbox id='uploadDefault' [extensionsAllow]= '.zip'></ej-uploadbox> | **Property**: allowedExtensions
 <ejs-uploader [allowedExtensions]= '.pdf'></ejs-uploader> |

| Restricted files types to select | **Property**: extensionsDeny
 <ej-uploadbox id='uploadDefault' [extensionsDeny]= '.docx'></ej-uploadbox> | **Not Applicable** |

| Display only selected details in File list | **Property** : customFileDetails
 <ej-uploadbox id='uploadDefault' [customFileDetails]= 'customDetails'></ej-uploadbox>
 TS:
public customDetails: any { title: false, name: true, size: true, status: true, action: false} | **Not Applicable** |

| Options to customize File list dialog | **Property**: dialogAction
 <ej-uploadbox [dialogAction]= 'dialogAction'></ej-uploadbox>
 TS:
 public dialogAction: any { modal: false, closeOnComplete: true, resize: true, drag: false, content: '#dialogTarget' } | **Not Applicable** |

| Customize dialog position | **Property:** dialogPosition
 <ej-uploadbox [dialogPosition]=
'position'></ej-uploadbox>
 TS:
 public position: object {X: 300, Y 100} | **Not Applicable** |

| Change file list key values | **Property:** dialogText
 <ej-uploadbox [dialogText]= 'dlgText'></ej-uploadbox>
 TS:
 public dlgText: { title: 'Upload File List', name: 'File Name', size: 'File Size' } | **Not Applicable** |

| Change drop area text | **Property:** dropAreaText
 <ej-uploadbox dropAreaText= 'Drop files here'></ej-uploadbox> | No separate Property to change dropAreaText. It can be customize using locale Texts |

| Change drop area height | **Property:** dropAreaHeight
 <ej-uploadbox [dropAreaHeight]=
'100%'></ej-uploadbox> | Not Applicable |

| Change drop area width | **Property:** dropAreaWidth
 <ej-uploadbox [dropAreaWidth]=
'100%'></ej-uploadbox> | Not Applicable |

| Dynamically push the file | **Property:** pushFile
 <ej-uploadbox [pushFile]= 'files'></ej-uploadbox> | Not Applicable |

| Show the files uploader in server already. | **Not Applicable** | **Property:** files
<ejs-uploader
[files]='preloadFiles'></ejs-uploader>
 TS:
 public preloadFiles: any = [{name: 'nature', size:
5000,type: '.png'}] |

| Event triggers when select the file successfully | **Event:** fileSelect
 <ej-uploadbox (fileSelect)=
'onFileSelect(\$event)'></ej-uploadbox>
 TS:
 public onFileSelect: any= () => { }; | **Event:**
selected
 <ejs-uploader (selected)=' onFileSelect(\$event)'></ejs-uploader>
 TS:

public onFileSelect: EmitType< SelectedEventArgs>= () => { }; |

Upload action

<!-- markdownlint-disable MD033 -->

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

| ----- | ----- | ----- |

| Save URL | **Property** : saveUrl
<ej-uploadbox [saveUrl]='saveUrl'></ej-uploadbox> |
Property : asyncSettings.saveUrl
<ejs-uploader [asyncSettings]='path'></ejs-uploader>

public path: Object = { saveUrl: 'https://ej2.syncfusion.com/services/api/uploadbox/Save'; } |

| Remove URL | **Property** : removeUrl
 <ej-uploadbox [removeUrl]=' removeUrl'></ej-uploadbox> | **Property** : asyncSettings.removeUrl
 <ejs-uploader [asyncSettings]='path'></ejs-uploader>
 public path: Object = { saveUrl:
'https://ej2.syncfusion.com/services/api/uploadbox/Save'; } |

| Automatically upload the file when files added in to upload queue | **Property:** autoUpload
<ej-uploadbox [autoUpload]='false'></ej-uploadbox> | **Property:** autoUpload
 <ejs-uploader [autoUpload]='false'></ejs-uploader> |

| Synchronous upload | **Property:** asyncUpload
 <ej-uploadbox [asyncUpload]='false'></ej-uploadbox> | No Separate Property for enabling synchronous upload. It can be enabling by using below configuration
<ejs-uploader [autoUpload]='false'></ejs-uploader> |

| Key to get the selected files in server side | **Property:** uploadName
 <ej-uploadbox [uploadName]='UploadKey'></ej-uploadbox> | Id of the element used as key value |

| Upload the files dynamically | **Not Applicable** | Method: upload()
 <ejs-uploader [autoUpload]='false' #upload></ejs-uploader>
 TS:
 @ViewChild('upload') public uploadObj: UploaderComponent; uploadObj.upload(filesData); |

| Event triggers before start to upload the action | **Event:** beforeSend
 <ej-uploadbox (beforeSend)= 'onBeforeSend (\$event)'></ej-uploadbox>
 TS:
 public onBeforeSend: any= () => { }; | **Event :** uploading
 <ejs-uploader (uploading)='beforeUploadStart (\$event)'></ejs-uploader>
 TS:
 public beforeUploadStart: EmitType<Object>= () => { }; |

| Event triggers when the upload is in progress | **Event:** inProgress
 <ej-uploadbox (inProgress)= 'uploadInProgress(\$event)'></ej-uploadbox>
 TS:
 public uploadInProgress: any= () => { }; | **Event :** progress
 <ejs-uploader #defaultupload (progress)='uploadInProgress (\$event)'></ejs-uploader>
 TS:
 public uploadInProgress: EmitType<Object>= () => { }; |

| Event triggers when upload got success | **Event:** success
 <ej-uploadbox (success)= 'uploadSuccess(\$event)'></ej-uploadbox>
 TS:
 public uploadSuccess: any= () => { }; | **Event :** success
 <ejs-uploader (success)='uploadSuccess(\$event)'></ejs-uploader>
 TS:
 public uploadSuccess: EmitType<Object>= () => { }; |

| Event triggers when upload got failed | **Event:** error
 <ej-uploadbox (error)= 'onUploadError(\$event)'></ej-uploadbox>
 TS:
public onUploadError: any= () => { }; | **Event :** failure
 <ejs-uploader (failure)='uploadFailure(\$event)'></ejs-uploader>
 TS:
 public uploadFailure: EmitType<Object>= () => { }; |

| Event triggers when the upload got started | **Event:** begin
<ej-uploadbox (begin)= 'onUploadBegin(\$event)'></ej-uploadbox>
 TS:
 public onUploadBegin: any= () => { }; | **Not Applicable** |

| Event triggers when cancel the upload | **Event:** cancel
 <ej-uploadbox (cancel)= 'onUploadCancel(\$event)'></ej-uploadbox>
 TS:
 public onUploadCancel: any= () => { }; | **Event :** canceling
 <ejs-uploader (canceling)='uploadingCancel(\$event)'></ejs-uploader>
 TS:
 public uploadingCancel: EmitType<Object>= () => { }; |

| Event triggers when the upload completed | **Event:** complete
 <ej-uploadbox (complete)= 'onUploadComplete(\$event)'></ej-uploadbox>
 TS:
 public onUploadComplete: any= () => { }; | **Not Applicable** |

Chunk Upload

<!-- markdownlint-disable MD033 -->

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

| ----- | ----- | ----- |

| Enabling the chunk upload | Not Applicable | **Property:** asyncSettings.chunkSize
 <ejs-uploader [asyncSettings]='path'></ejs-uploader>
 TS:
 public path: Object = { saveUrl: 'saveUrl',chunkSize: 50000 }; |

| Retry the upload automatically when it's get failed | Not Applicable | **Property:** asyncSettings.retryCount, asyncSettings.retryAfterDelay
 <ejs-uploader [asyncSettings]='path'></ejs-uploader>
 TS:
 public path: Object = { saveUrl: 'saveUrl',chunkSize: 50000,retryCount: 3,retryAfterDelay: 1000}; |

| Pause the uploading file | Not Applicable | **Method:** pause
 <ejs-uploader #defaultupload></ejs-uploader>
 TS:
 @ViewChild('defaultupload') public uploadObj: UploaderComponent; uploadObj.pause(filesData); |

| Event triggers when pausing the file | Not Applicable | **Event:** pausing
 <ejs-uploader (pausing)='pausingUpload(\$event)'></ejs-uploader>
 TS:
 public pausingUpload: EmitType<Object>= () => { }; |

| Resuming the paused file | Not Applicable | **Method:** resume
 <ejs-uploader #defaultupload></ejs-uploader>
 TS:
 @ViewChild('defaultupload') public uploadObj: UploaderComponent; uploadObj.resume(filesData); |

| Event triggers when resuming the file | Not Applicable | **Event:** resuming
 <ejs-uploader (resuming)='resumingUpload(\$event)'></ejs-uploader>
 TS:
 public resumingUpload: EmitType<Object>= () => { }; |

| Retry the failed file | Not Applicable | **Method:** retry
 <ejs-uploader #defaultupload></ejs-uploader>
 TS:
@ViewChild('defaultupload') public uploadObj: UploaderComponent;uploadObj.retry(filesData); |

| Cancel the failed file | Not Applicable | **Method:** cancel
 <ejs-uploader #defaultupload></ejs-uploader>
 TS:
 @ViewChild('defaultupload') public uploadObj: UploaderComponent; uploadObj.cancel(filesData); |

| Event triggers when cancel the file | Not Applicable | **Event:** canceling
 <ejs-uploader (canceling)='cancelingUpload(\$event)'></ejs-uploader>
 TS:
 public cancelingUpload: EmitType<Object>= () => { }; |

| Event triggers when chunk file fails | Not Applicable | **Event:** chunkFailure
 <ejs-uploader (chunkFailure)='onChunkFailure(\$event)'></ejs-uploader>
 TS:
 public onChunkFailure: EmitType<Object>= () => { }; |

| Event triggers when chunk file success | Not Applicable | **Event:** chunkSuccess
 <ejs-uploader (chunkSuccess)='onChunkSuccess(\$event)'></ejs-uploader>
 TS:
 public onChunkSuccess: EmitType<Object>= () => { }; |

Remove action

<!-- markdownlint-disable MD033 -->

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

| ----- | ----- | ----- |

| Remove the uploaded file | Not Applicable | **Method:** remove
 <ejs-uploader #defaultupload></ejs-uploader>
 TS:
 @ViewChild('defaultupload') public uploadObj: UploaderComponent; uploadObj.remove(filesData); |

| Event triggers when the file removing succeed | **Event:** remove
 <ej-uploadbox (remove)='onRemove(\$event)'></ej-uploadbox>
 TS:
 public onRemove: any= () => { }; | **Event:** success
 <ejs-uploader (success)='onSuccess(\$event)'></ejs-uploader>
 TS:
 public onSuccess: EmitType<Object>= () => { }; |

| Event triggers when the file removing fails | **Not Applicable** | **Event:** failure
 <ejs-uploader (failure)='onFailure(\$event)'></ejs-uploader>
 TS:
 public onFailure: EmitType<Object>= () => { }; |

Buttons

<!-- markdownlint-disable MD033 -->

| **Behavior** | **Property in essential JS 1** | **Property in essential JS 2** |

| ----- | ----- | ----- |

| Customize button text | **Property** : buttonText
 <ej-uploadbox buttonText.browse]='browse' [buttonText.upload]='upload' [buttonText.cancel]='cancel'></ej-uploadbox>
 TS:
 public browse: string = 'Choose File'; public upload: string = 'Upload File'; public cancel: string = 'Cancel Upload'; | **Property** : buttons
 <ejs-uploader [buttons]='buttons'></ejs-uploader>
 TS:
 public buttons: object { browse: 'Choose File', clear: 'Clear files', upload: 'upload Files' } |

Drag and Drop

<!-- markdownlint-disable MD033 -->

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

| ----- | ----- | ----- |

| Enable drag and drop upload | **Property** : allowDragAndDrop
 <ej-uploadbox [allowDragAndDrop]='true'></ej-uploadbox> | No separate Property to disabling drag and drop |

| Set custom drop area | **Not Applicable** | **Property** : dropArea
 <ejs-uploader [dropArea]='dropElement'></ejs-uploader> |

Common

<!-- markdownlint-disable MD033 -->

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

| ----- | ----- | ----- |

| Adding custom class to wrapper element | **Property** : cssClass
 <ej-uploadbox cssClass='Custom-Class'></ej-uploadbox> | Not Applicable |

| Enable/Disable the control | **Property** : enabled
 <ej-uploadbox id='uploadBox' [enabled]='false'></ej-uploadbox> **Method** : enable(), disable()
 \$('#uploadBox').ejUploadbox('enable'); \$('#uploadBox').ejUploadbox('disable'); | **Property:** enabled
 <ejs-uploader [enabled]='false'></ejs-uploader> |

| Set height for uploader | **Property:** height
 <ej-uploadbox id='uploadBox' height='100%'></ej-uploadbox> | **Not Applicable** |

| Set width for uploader | **Property:** width
 <ej-uploadbox id='uploadBox' width='100%'></ej-uploadbox> | **Not Applicable** |

| Adding HTML attributes | **Property:** htmlAttributes
 <ej-uploadbox id='uploadBox' [htmlAttributes]='htmlAttribute'></ej-uploadbox>
 TS:
 public htmlAttribute: object = { 'aria-label': 'UploadBox'} | **Not Applicable** |

| Event triggers when control created successfully | **Event:** create
 <ej-uploadbox (create)= 'onCreate(\$event)'></ej-uploadbox>
 TS:
 public onCreate: any= () => { }; | **Event:** created
 <ejs-uploader (created)= 'onCreated(\$event)'></ejs-uploader>
 TS:
 public onCreated: EmitType<Object>= () => { }; |

| Event triggers when destroy the control | **Event:** destroy
 <ej-uploadbox (destroy)= 'onDestroy(\$event)'></ej-uploadbox>
 TS:
 public onDestroy: any= () => { }; | **Not Applicable** |

| Keeping the model values in cookies | **Property:** enablePersistence
 <ej-uploadbox id='uploadBox' [enablePersistence]='true'></ej-uploadbox> | **Property:** enablePersistence
 <ejs-uploader [enablePersistence]='true'></ejs-uploader> |

| Get the selected files data | **Not Applicable** | **Method:** getFilesData
 <ejs-uploader #defaultupload></ejs-uploader>
 TS:
 @ViewChild('defaultupload') public uploadObj: UploaderComponent; uploadObj.getFilesData(); |

| Convert bytes in to MB, GB | **Not Applicable** | **Method:** bytesToSize
 <ejs-uploader #defaultupload></ejs-uploader>
 TS:
 @ViewChild('defaultupload') public uploadObj: UploaderComponent; uploadObj.bytesToSize(5000); |