Jasmine

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia https://twitter.com/tutorialspoint

About the Tutorial

Jasmine is one of the most popular tools for a JavaScript developer to deal with hectic testing process. It is an open source technology. It is a simple API to test different components of JavaScript.

This tutorial discusses the basic functionalities of Jasmine.js along with relevant examples for easy understanding.

Audience

This tutorial has been prepared for beginners to help them understand the basic concepts of Jasmine testing process.

Prerequisites

This is a very basic tutorial and it should be useful to any reader with a reasonable knowledge of basic computer programming.

Copyright & Disclaimer

Copyright 2016 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

tutorialspoint
SIMPLYEASYLEARNING

Table of Contents

tutorialspoint
SIMPLYEASYLEARNING

tutorialspoint
SIMPLYEASYLEARNING

tutorialspoint
SIMPLYEASYLEARNING

1. Jasmine – Overview

Jasmine is an open-source JavaScript framework, capable of testing any kind of JavaScript application. Jasmine follows Behavior Driven Development (BDD) procedure to ensure that each line of JavaScript statement is properly unit tested. By following BDD procedure, Jasmine provides a small syntax to test the smallest unit of the entire application instead of testing it as a whole.

Why Use Jasmine?

Following are the advantages of using Jasmine over other available JavaScript testing frameworks:

- Jasmine does not depend on any other JavaScript framework.
- Jasmine does not require any DOM.
- All the syntax used in Jasmine framework is clean and obvious.
- Jasmine is heavily influenced by Rspec, JS Spec, and Jspec.
- Jasmine is an open-source framework and easily available in different versions like stand-alone, ruby gem, Node.js, etc.

How to Use Jasmine?

Jasmine is very easy to implement in any kind of development methodology. All you need to download is the standalone library files from the official website http://jasmine.github.io/ and implement the same in your application.

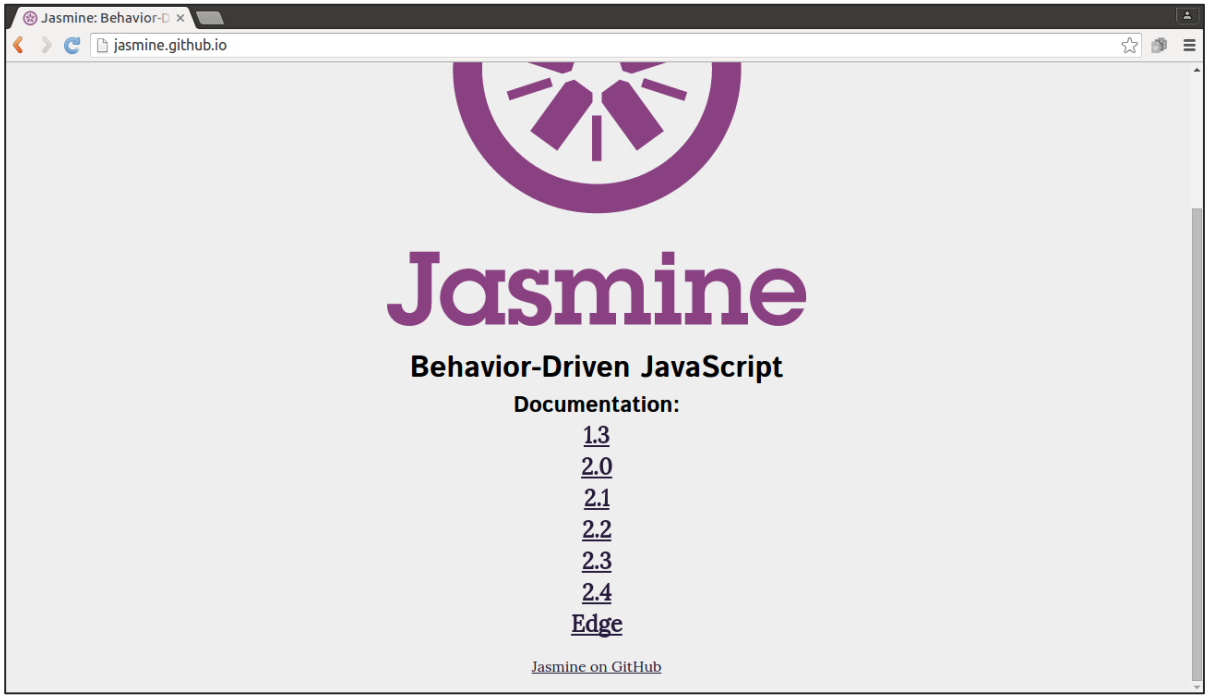
The detailed environment setup will be described in the next chapter named "Environment setup". Once you successfully download and unzip the zip file, then you will find the following sub-folders inside that zip file.

tutorialspoint
SIMPLYEASYLEARNING

2. Jasmine – Environment Setup

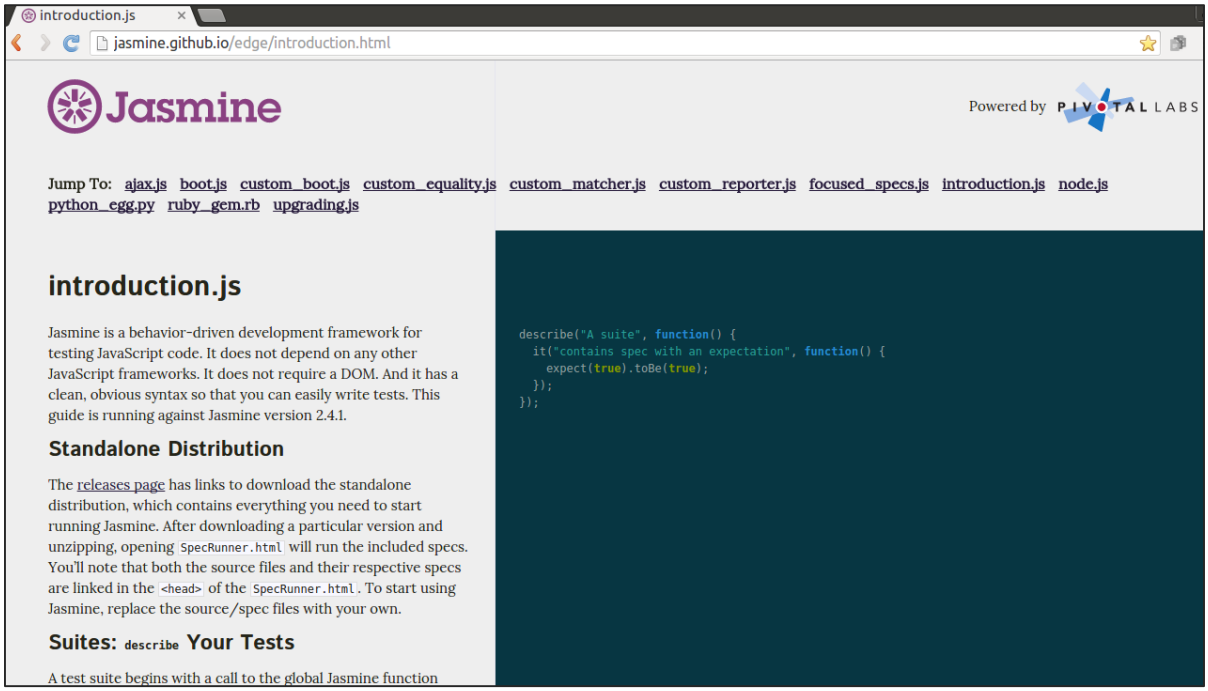
In this chapter, we will discuss the step-by-step procedure of how to set up a Jasmine-based BDD testing application.

Step 1: Go to the official website of jasmine http://jasmine.github.io/

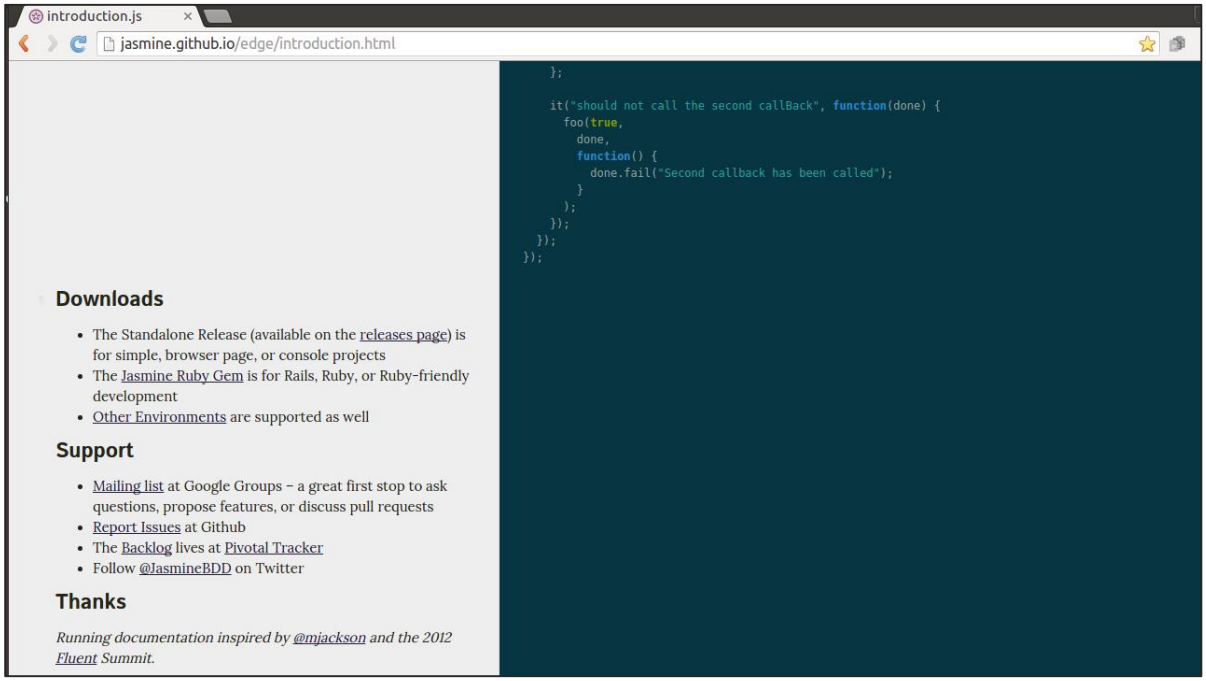


Step 2: Click on any of the version link. It is preferable to use the most recent version that is "Edge". You will be redirected to the homepage of the selected version.

tutorialspoint
SIMPLYEASYLEARNING

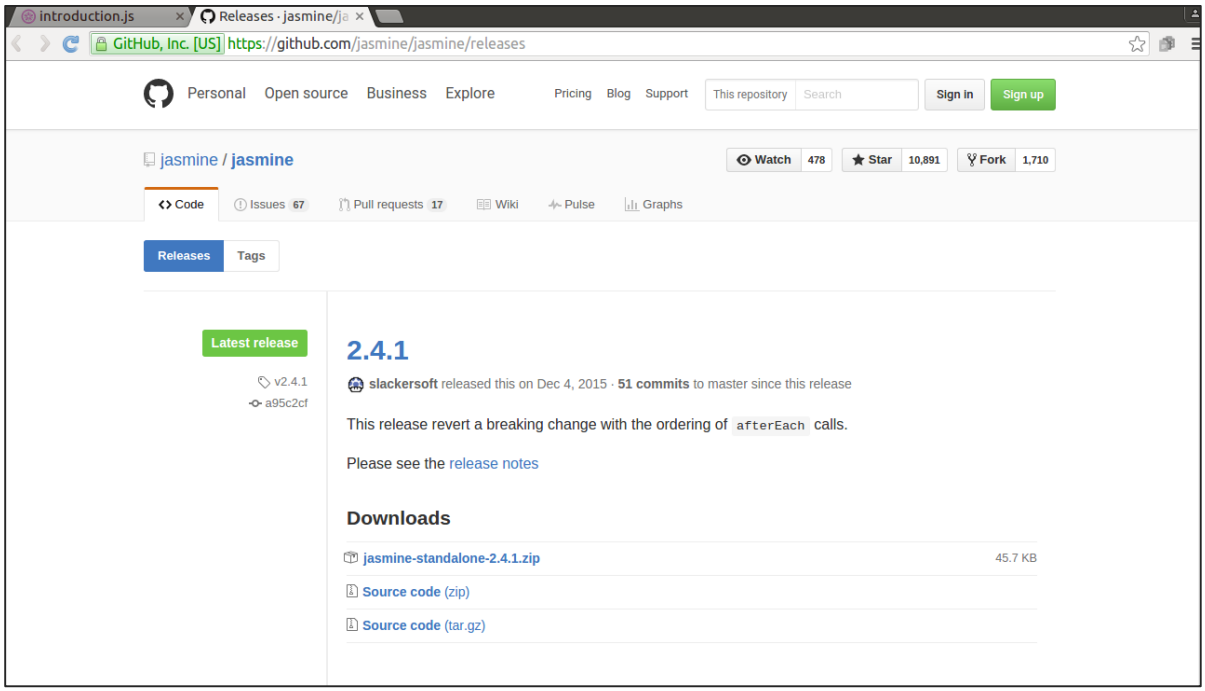


Step 3: Go to the Download section of the homepage and click on the standalone release page.



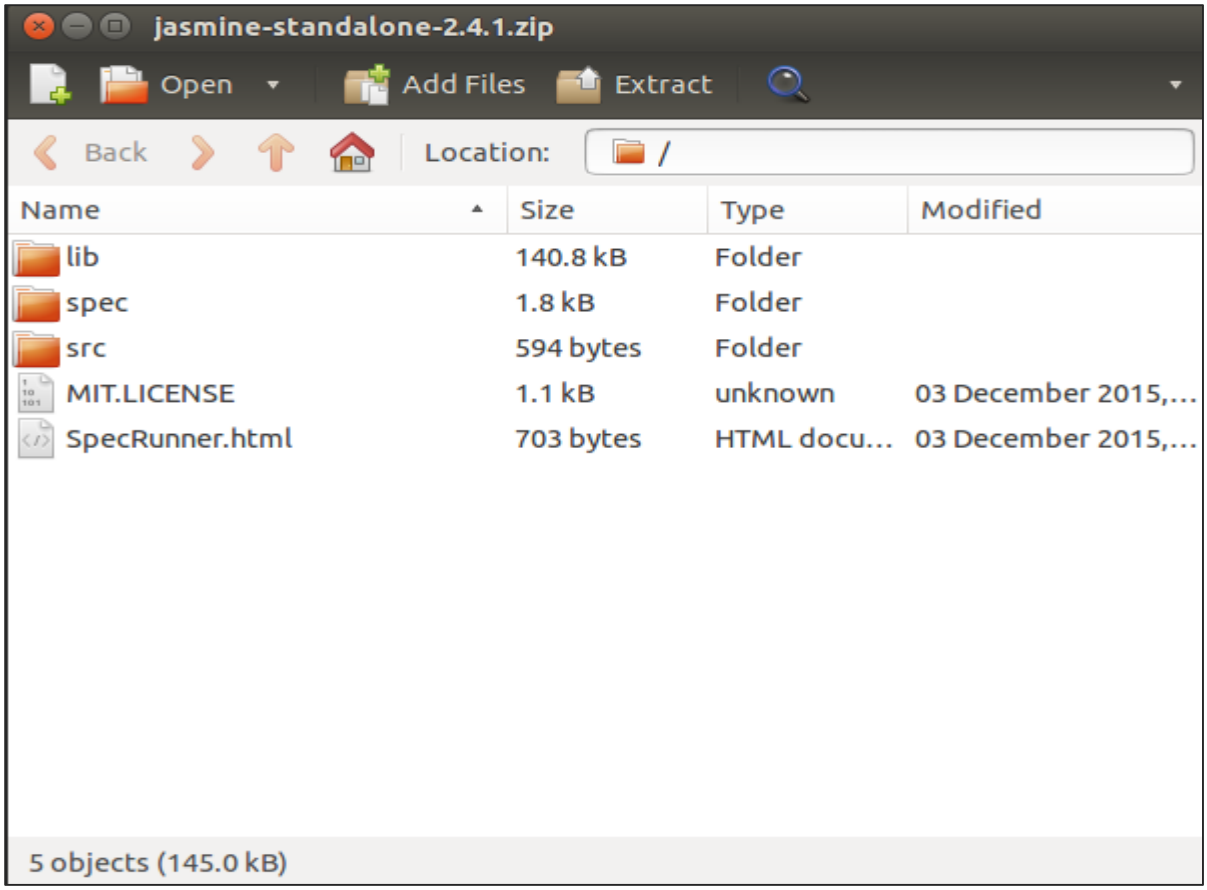
tutorialspoint
SIMPLYEASYLEARNING

Step 4: Once you are redirected to github release page, download the Zip file from there.

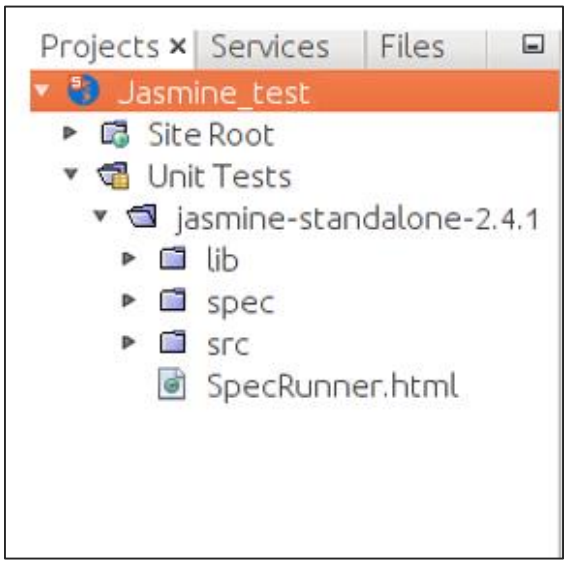


Step 5: Unzip the downloaded jasmine-standalone-2.4.1 folder. You will see the following folder structure.

tutorialspoint
SIMPLYEASYLEARNING



Step 6: Now Create a web application project in your favorite IDE and add this downloaded library files into the application. Here, we have used netbeans IDE. Following is the Directory structure of our application after adding Jasmine framework.



Our environment setup is done. Now our application is ready to be tested by Jasmine framework.

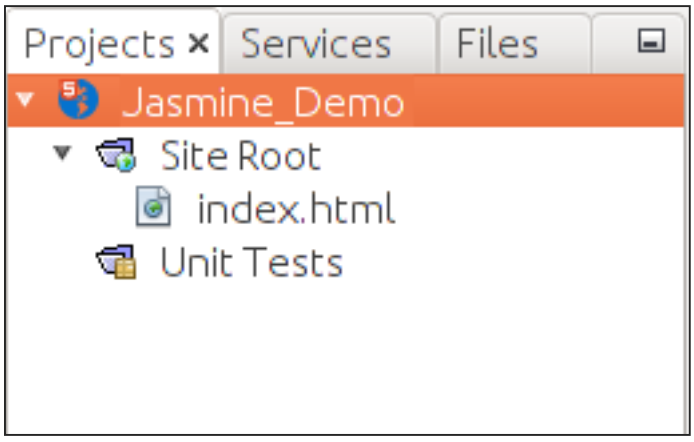
tutorialspoint
SIMPLYEASYLEARNING

3. Jasmine – Writing Test & Execution

In this chapter, we will create a **hello world app** which will test our "**helloworld.js**" file. Before developing the hello world app, go back to the previous chapter and make sure that your environment is ready to be tested using Jasmine.

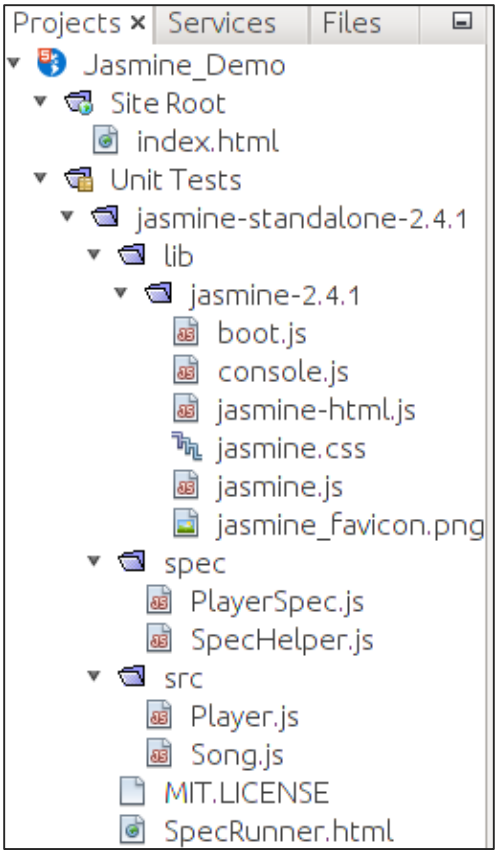
Step 1: Create a Web application in your IDE

Here we are using NetBeans 8.1 to develop our hello world app in Jasmine. In NetBeans, go to File -> New Project ->Html5/JS application and create a project. After creating the project, the project directory should look like the following screenshot. We named our project as **Jasmine_Demo**.



Step 2: Include the Jasmine lib file into the application

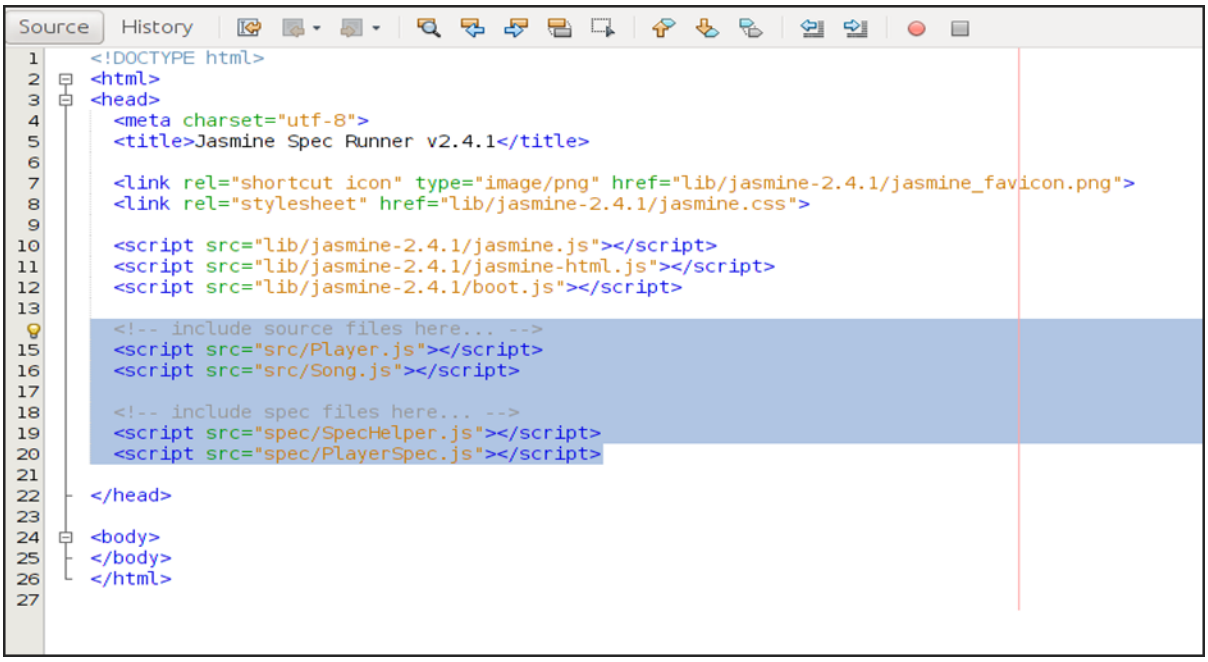
After creating the demo project all you need to do is include the unzip folder of Jasmine library in the Unit Tests folder of the created application. After adding all the library files to our application folder, the structure of our project will look like as shown in the following screenshot.



Files given under **spec** and **src** folders are demo files provided by the Jasmine team. Delete these files as we are going to create our own test file and test case. While deleting those JavaScript file, we need to delete the reference of those files inside our output html file that is **SpecRunner.html**.

Following is the screenshot of SpecRunner.html file where the reference of different JavaScript files inside **spec** and **src** will be deleted.

tutorialspoint
SIMPLYEASYLEARNING



Step 3: Create a JavaScript file

In this step, we will create a JavaScript file named **helloworld.js** under **src** folder. This is the file which we will test through Jasmine. After creating a JavaScript file append the following set of code inside the file.

```
/*
* This is the JavaScript file that need to be tested through jasmine
* Below is the helloworld function that will return 'Hello World'
*
*/
var helloworld = function(){
     return 'Hello World';
};
```

Step 4: Create a test case

In this step, we will create another JavaScript file which will contain the test case for the above-mentioned JavaScript file. Go ahead and create a JavaScript file under "Spec" folder and name it as **"HelloWorldsSpec.js"**. Add the following line of code into this **js** file.

```
/*
* This is the file which will call our java script file that need to be tested. *
Each describe block is equivalent to one test case
```

```
*
*/
describe("Hello World", function(){
     it("should Return Hello world",function(){
         expect(helloworld()).toEqual('Hello World');
     });
});
```

Step 5: Add reference to the output file

We successfully created our own file to be tested and the corresponding test case. We kept it under two different folders. In this step, we will modify **"SpecRunner.html"** to include the reference of these two newly created file.

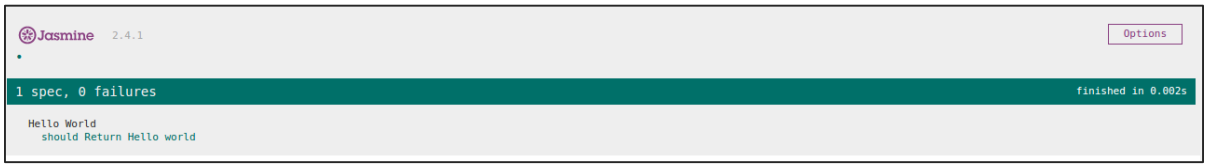
```
<!DOCTYPE html>

<html>
<head>
<meta charset="utf-8">

<title>Jasmine Spec Runner v2.4.1</title>
<link rel="shortcut icon" type="image/png" href="lib/jasmine-
2.4.1/jasmine_favicon.png">
<link rel="stylesheet" href="lib/jasmine-2.4.1/jasmine.css">
<script src="lib/jasmine-2.4.1/jasmine.js"></script>
<script     src="lib/jasmine-2.4.1/jasmine-html.js"></script>
<script  src="lib/jasmine-2.4.1/boot.js"></script>
<!--Lines to be deleted
<script  src="src/Player.js"></script>
<script  src="src/Song.js"></script>
<script src="spec/SpecHelper.js"></script>
<script src="spec/PlayerSpec.js"></script>-->
```


tutorialspoint
SIMPLYEASYLEARNING

```
<!--adding the reference of our newly created file --->
<script src="src/helloworld.js"></script>
<script  src="spec/HelloWorldsSpec.js"></script>
</head>
<body>
</body>
</html>
```

Step 6: Execute by running SpecRunner.html

This is the final step of our application development. Run SpecRunner.html in any of your favorite browser. The following screenshot will appear as a result. The green screen indicates success, whereas red indicates failure in test case.



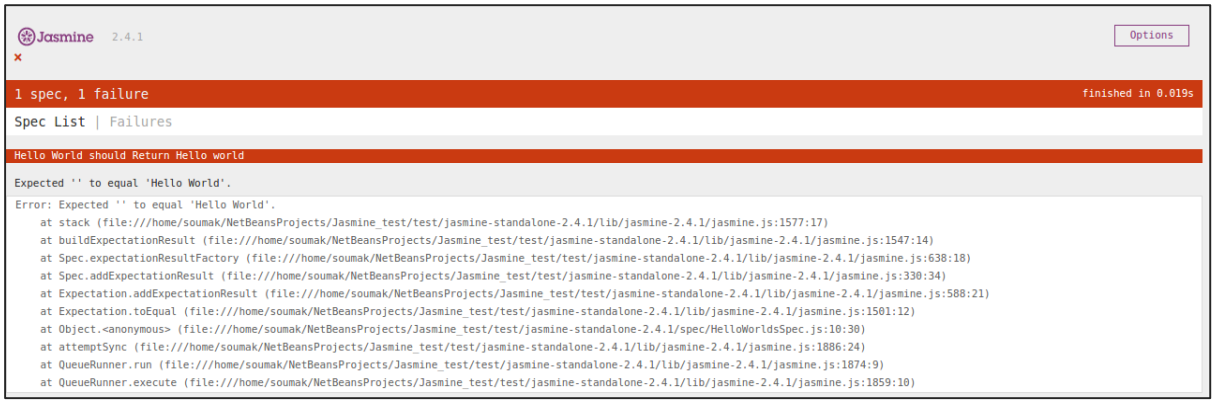
Step 7: Understand the failure case

Till now we have seen the **success** test case of the hello world application. Now let us see what if something goes wrong and the test fails. To implement a failure case we need to write a failure test case. To do the same, we are going to modify the **helloworld.js** file using the following code.

```
var helloworld = function (){
     return '';
};
// we are not returning any string whereas in the spec file we are expecting a //
string as “Hello World”
```

The above code is definitely going to fail because our spec file is not getting the expected string as an output of the **helloworld()**. The following screenshot of the **specRunner.html** file depicts that there is an error with its red indicator.

tutorialspoint
SIMPLYEASYLEARNING

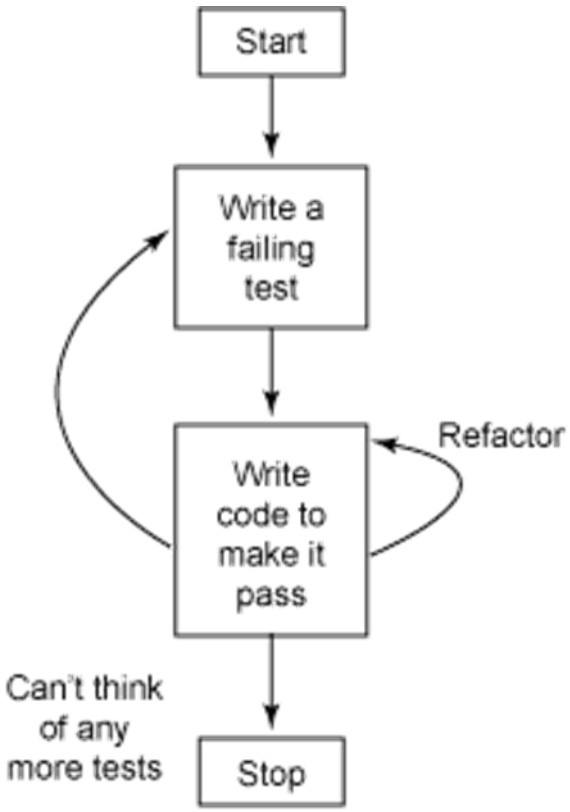


tutorialspoint
SIMPLYEASYLEARNING

4. Jasmine – BDD Architecture

Jasmine follows the Behavioral Driven Development (BDD) framework. Before learning the working principle of Jasmine, let us know what is the BDD framework.

The following flowchart depicts the different phases of BDD framework.



Step 1: Start

In this phase, we will make our environment ready for Jasmine application.

Step 2: Write a failing test

In this step, we will write our first ever test case. It is obvious that this test is going to fail because there is no such file or function to be tested.

tutorialspoint
SIMPLYEASYLEARNING

Step 3: Write a code to make it pass

In this phase, we will prepare our JavaScript file or function that needs to be tested. This phase is crucial as we need to make sure that all the test cases we had prepared in the early stage will be successful.

Step 4: Refactor

Refactor is a very important phase of BDD model where we need to prepare as many test cases as we can for that particular application or function.

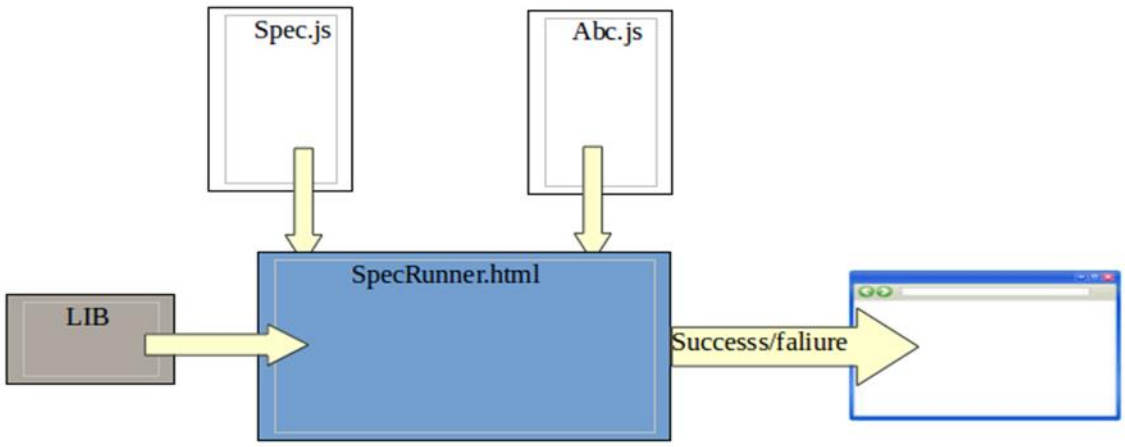
Step 5: Stop

If everything is going well then your application must be ready and up. So we can consider this step as an end of our BDD application.

Example

We have now gathered some knowledge about the working principle of BDD framework. Let us see how Jasmine follows this BDD framework in the line of JavaScript testing.

As the screenshot depicts we need to test Abc.js using Jasmine framework. **SpecRunner.html** is the output file that will take **Spec.js(Test case file)**, **Abc.js(file to be tested)**, LIB as an input and run all the test cases present in the spec file and render the result into the browser.



Lib: These are the inbuilt JavaScript files that will be helpful to test different functions and other JavaScript files used in our project.

Spec.js(Test case file): This is the JavaScript file that contains all the test cases which is required to test any JavaScript function or file. In the BDD methodology, we are going to write the test first, hence this is the file that needs to be updated first. Definitely this is going to be

tutorialspoint
SIMPLYEASYLEARNING

fail as there is no such file or function present in our project that can be tested. This file can be refactored unlimited times until all the functionalities are tested.

Abc.js(File to be tested): This is the file that contains your functionalities which will be unit tested using Spec.js and Lib file.

SpecRunner.html: SpecRunner.html is a normal html file which will render the output of the unit test with the help of embedded JavaScript codes in it.

tutorialspoint
SIMPLYEASYLEARNING

End of ebook preview

If you liked what you saw…

Buy it from our store @ **https://store.tutorialspoint.com**

tutorialspoint
SIMPLYEASYLEARNING