About Jasmine

Jasmine is behavior-driven development framework for testing JavaScript code. It does not depend on any other JavaScript frameworks. It does not require a DOM. And it has a clean, obvious syntax so that you can easily write tests. Its a Formal Definition, you'll find everywhere.

Main thing what it actually does and how. Lets start

Following steps to create Jasmine test cases:

- You have to create a html file name i.e. specRunner.html which has reference of script files which
- has sequence, not sequence actually according to their dependency to different module. we use direct cdn(*content distribution network*) of references or download it in your project.
-

```
<link rel="shortcut icon" type="image/png" href="jasmine/lib/jasmine-2.0.0/jasmine_favicon.png">
<link rel="stylesheet" type="text/css" href="jasmine/lib/jasmine-2.0.0/jasmine.css">

<script type="text/javascript" src="jasmine/lib/jasmine-2.0.0/jasmine.js"></script>
<script type="text/javascript" src="jasmine/lib/jasmine-2.0.0/jasmine-html.js"></script>
<script type="text/javascript" src="jasmine/lib/jasmine-2.0.0/boot.js">
</script><script src="Lib/angular/angular.js"></script>
<script src="Lib/angular-mocks/angular-mocks.js"></script>
<script src="Lib/jquery/dist/jquery.js"></script>
```

- After that you have to add your angular module files, controller js files in html file.
- Now create a test_spec.js file which has jasmine code inside it . also add it's reference in specRunner.html , Before creating test_spec you must be aware of its syntax and how it works.

Jasmine code syntax

```
describe('JavaScript addition operator', function () {
    it('adds two numbers together', function () {
        expect(1 + 2).toEqual(3);
    });
});
```

Both the describe and it functions take two parameters: a text string and a function. Most test frameworks try to read as much like English as possible, and you can see this with Jasmine. First, notice that the string passed to describe and the string passed to it form a sentence (of sorts): "JavaScript addition operator adds two numbers together." Then, we go on to show how.

Inside that it block, you can write all the setup code you need for your test. We don't need any for this simple example. Once you're ready to write the actual test code, you'll start with the expect function, passing it whatever you are testing. Notice how this forms a sentence as well: we "expect 1 + 2 to equal 3."

```
//This won't be run if there are specs using describe only or // it only
describe("Spec 2", function(){})
```

Since describe and it blocks are functions, they can contain any executable code necessary to implement the test. JavaScript scoping rules apply, so variables declared in a describe are available to any it block inside the suite.

- For expectations about the state of the DOM, Jasmine uses jasmine-jquery matcher and also uses its predefined Matchers. It offers a ton of Matchers. some of them are below:

```
 expect(true).toBe(true);
 expect("ABCDEF").toContain("ABC");
 expect(123).toEqual(123);
 expect().toHaveBeenCalled();
 expect().toBeDefined();
```

There's a lot more you can do with Jasmine: function-related matchers, spies, asynchronous specs, and more.
So if you aren't testing your JavaScript so far, now is an excellent time to start. As we've seen, Jasmine's fast and simple syntax makes testing pretty simple.

Test Case Using Jasmine

Simple Test Case

Angular Code

```
 var app=angular.module("Test Demo");
app.controller("Test_Controller",['$scope','$rootScope','$http',
function ($scope,rootScope,$http){
var vm = this;
vm.Name="test_abc";
vm.Age=20;
});
```

Jasmine Test Case

```
describe("Advanced Search ", function () {
var scope,controller;
// Inject Particular Module
 beforeEach(angular.mock.module("Test Demo""));
// Injecting Controller , inject is like constructor where you   // have to define your dependencies which you have to use, just  // like if you want to inject controller,first define it then you// through $controller(controllername). There's no boundation    // that you have to use it under beforeEach you can also use it  // inside 'it' function, it would reduce execution timing.
beforeEach(inject(function ($rootScope, $injector, $controller) { scope = $injector.get('$rootScope');
controller = $controller('Test_Controller', { '$scope': scope });}));
it('Should Initialize controller', function () {
expect(controller).toBeDefined(); });
});
it('Should Check Vales of controller Variables', function () {
expect(controller.Name).toBe("test_abc"); });
});
it('Should Check Vales of controller Variables', function () {
expect(controller.Age).toBe(20); });
});
//Here You would ask about beforeeach, beforeeach is nothing it  // will excute before every 'it' function.
```