Jasmine is an open-source and most popular JavaScript library testing framework to test any kind of JavaScript application. Jasmine follows Behavior Driven Development (BDD) procedure to ensure that each line of JavaScript statement is properly unit tested.

Installation:

Need node js

npm install -g jasmine-node

jasmine init

The above commands will create a package.json and spec folder where we need to write our spec files.

In the all the examples below,

Create a folder under specs called Examples.

And keep the corresponding spec files under spec directly.

Example 1:

**Calculator.js**

window.Calculator = {

currentVal:0,

varAfterEachExmaple:0,

add:function (num1) {

this.currentVal += num1;

return this.currentVal;

},

addAny:function () {

var sum = this.currentVal;

for(var i = 0; i < arguments.length; i++) {

sum += arguments[i];

}

this.currentVal = sum;

Return this.currentVal;

},

};

**CalculatorSpec.js**

describe("calculator",function() {

//test case: 1

it("Should retain the current value of all time", function () {

expect(Calculator.currentVal).toBeDefined();

expect(Calculator.currentVal).toEqual(0);

});

//test case: 2

it("should add numbers",function() {

expect(Calculator.add(5)).toEqual(5);

expect(Calculator.add(5)).toEqual(10);

});

//test case :3

it("Should add any number of numbers",function () {

expect(Calculator.addAny(1,2,3)).toEqual(6);

});

});

Example 2:

describe('This custom matcher example', function() {

beforeEach(function() {

// We should add custom matched in beforeEach() function.

jasmine.addMatchers ({

validateAge: function() {

Return {

compare: function(actual,expected) {

var result = {};

result.pass = (actual > = 13 && actual < = 19);

result.message = 'sorry u are not a teen ';

return result;

}

};

}

});

});

it('Lets see whether u are teen or not', function() {

var myAge = 14;

expect(myAge).validateAge();

});

it('Lets see whether u are teen or not ', function() {

var yourAge = 18;

expect(yourAge).validateAge();

});

});

Example 3:

var currentVal = 0;

beforeEach(function() {

currentVal = 5;

});

describe("Different Methods of Expect Block",function() {

it("after each function ", function() {

expect(currentVal).toEqual(5);

});

});

Example 4:

var myCustomMatchers = {

// toBeAllowedToDrive matcher

// Usage: expect(age).toBeAllowedToDrive();

// expect(age).not.toBeAllowedToDrive();

toBeAllowedToDrive: function() {

return {

compare: function(age) {

var result = {};

result.pass = age>16;

if (result.pass) {

result.message = "Expected " + age + " to be allowed to drive";

} else {

result.message = "Expected " + age + " to be allowed to drive, but it was not";

}

return result;

}

};

}

};

describe("Custom matcher: 'toBeAllowedToDrive'", function() {

var John = 17,

Mary = 16;

// Custom Matchers must be added using beforeEach

beforeEach(function() {

jasmine.addMatchers(myCustomMatchers);

});

it("should allow John to drive", function() {

expect(John).toBeAllowedToDrive();

// replaces

expect(John).toBeGreaterThan(16);

});

it("should not allow Mary to drive", function() {

expect(Mary).not.toBeAllowedToDrive();

// replaces

expect(Mary).not.toBeGreaterThan(16);

});

});