EBook Gratis

APRENDIZAJE jasmine

Free unaffiliated eBook created from
Stack Overflow contributors.

#jasmine

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: jasmine

It is an unofficial and free jasmine ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official jasmine.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con el jazmín

Observaciones

En algún momento, probar nuestro código JavaScript se convierte en una tarea difícil. Jasmine es un marco de desarrollo basado en el comportamiento para probar nuestro código JavaScript. No depende de ningún otro framework de JavaScript. No requiere un DOM. Y tiene una sintaxis limpia que te hace escribir fácilmente las pruebas. Puede encontrar la documentación de Jasmine aquí y el proyecto en GitHub .

Versiones

Versión	Fecha de lanzamiento
1.0.0	2010-09-14
1.3.0	2012-11-27
2.0.0	2013-12-16
2.1.0	2014-11-14
2.2.0	2015-02-02
2.3.0	2015-04-28
2.4.0	2015-12-02
2.5.0	2016-08-30

Examples

Instalación o configuración

Instalación de jazmín independiente

Descarga la última versión de Jasmine de la página de lanzamiento de Jasmine :

Ejecutando Jasmine localmente

1. Ejecute Jasmine en el navegador descargando el archivo zip, extrayéndolo, haciendo referencia a los archivos de la siguiente manera:

```
<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>
```

Instalación de Jasmine usando npm (Administrador de paquetes de nodos)

1. Configurar directorio de proyectos para Jasmine

Cree una carpeta y ejecute `npm init` esto creará un archivo `package.json` vacío y le hará algunas preguntas sobre su proyecto para llenar el archivo `json` proyecto.

Agregar 2 `app` directorios - para el servidor y `spec` - para pruebas

2. Conseguir jazmín

   Desde el directorio raíz del proyecto ejecute

   ```
   npm install jasmine-node --save

   npm install request --save

   npm install express --save
   ```

   esto te conseguirá los paquetes

   `./node_packages/.bin/jasmine-node spec` ejecutará jasmine binary

   Después de esto, tu `package.json` debería verse similar a este

   archivo package.json, después de lo cual ese archivo debería verse así:

```
{
    "name": "Jasmine",
    "version": "0.0.1",
    "description": "Jasmine",
    "main": "index.js",
    "scripts": {
      "test": "./node_modules/.bin/jasmine-node spec"
    },
    "author": "Me",
    "license": "ISC"
  }
```

Instalar con npm

```
npm install -g jasmine
```

Si está siendo usado con karma, instale `karma-jasmine`

```
npm install --save-dev karma-jasmine
```

Hola Mundo

Para crear una prueba más básica con Jasmine, vaya a la carpeta de `spec` (pruebas) y agregue el archivo llamado `testSpec.js` .

En ese archivo agregue lo siguiente:

```
var request = require("request");

describe("Hello World Test", function() {
```

```
  // This is your test bundle

  describe("GET SO", function() {
    //This is testing that http GET works

    it("Checks if SO is online", function() {
      // This is description of your test - this is what you get when it fails

      request.get("http://stackoverflow.com/", function(error, response, body) {
        // this is your test body

        expect(response.statusCode).toBe(200);
        // this is your test assertion - it expects status code to be '200'
      });
    });
  });
});
```

Lea Empezando con el jazmín en línea: https://riptutorial.com/es/jasmine/topic/1302/empezando-con-el-jazmin

Capítulo 2: Atributos

Observaciones

Hay algunos términos que debe tener en cuenta antes de escribir los casos de prueba de Jasmine.

1. Suites

Un traje es el punto de partida de los casos de prueba de Jasmine, en realidad se denomina descripción de la función jazmín global. Puede tener dos parámetros, un valor de cadena que describe la demanda y una función que implementa la demanda.

2. Especulación

Al igual que las suites, una especificación comienza con una cadena que puede ser el título de la demanda y una función donde escribimos las pruebas. Una especificación puede contener una o más expectativas que prueben el estado de nuestro código.

3. Expectativa

El valor de una expectativa es verdadero o falso, una expectativa comienza con la función esperar. Toma un valor y llama al real.

Examples

Suites

```
describe("Includes validations for index page", function () {

});
```

Especulación

```
it("Spy call for datepicker date validation", function () {

});
```

Expectativa

```
describe("Includes validations for index page", function () {
    var indexPage;

    it("Check for null values", function () {
        // We are going to pass "" (null) value to the function
        var retVal = indexPage.isNullValue("");
        expect(retVal).toBeTruthy();
```

```
    });

});
```

Lea Atributos en línea: https://riptutorial.com/es/jasmine/topic/7980/atributos

Capítulo 3: Espías

Observaciones

Un espía se define como una función específica de prueba que intercepta llamadas a una función subyacente en el código de la aplicación y despacha su propia implementación cuando se llama a la función subyacente para probar la interfaz en lugar de la implementación.

Examples

Espiando una función existente

Jasmine puede espiar una función existente usando la función `spyOn` .

```
let calculator = {
  multiply: function(a, b) {
     return a * b;
  },

  square: function(a) {
    return this.multiply(a, a);
  }
}


describe('calculator', function() {
  it('squares numbers by multiplying them by themselves', function() {
    let num = 2;
    spyOn(calculator, 'multiply');
    calculator.square(NUM);
    expect(calculator.multiply).toHaveBeenCalledWith(NUM, NUM);
  })
});
```

Una vez que se ha espiado la función, se reemplaza por un espía, que se puede consultar para obtener información sobre cómo y cuándo se ha llamado.

Creando un nuevo espia

Podemos usar `jasmine.createSpy()` para crear un espía independiente. Esto suele ser útil si necesitamos pasar una función como devolución de llamada a otra función y queremos probar cómo se usa.

```
// source code
function each(arr, fn) {
    arr.forEach(fn);
}

// test code
describe('each', function() {
```

```
  let mockFn = jasmine.createSpy();

  it('calls a function for each item in the array ', function() {
    let arr = [1,2,3,4,5]
    each(arr, mockFn);
    expect(mockFn.calls.count()).toBe(arr.length);
  })
});
```

Espiando en un servicio angular.

En este ejemplo tenemos un servicio, llamémoslo servicio de búsqueda que tiene un método llamado search () que iniciará una solicitud de obtención a una API de back-end.

```
function SearchService($http) {
    const service = {};

    service.search = function() {
        return $http({method: 'GET', url: `/api/search`})
    }

    return service;
}
angular.module('app').factory('searchService', SearchService);
```

Pruebas

```
describe('search service', function() {
    var $httpBackend;
    var searchService;
    beforeEach(angular.mock.module('app'));

    beforeEach(inject(function(_$httpBackend_, _searchService_) {
        $httpBackend = _$httpBackend_;
        searchService = _searchService_;
    }));

    it('should perform http call to the search api', function(){
        searchService.search();
        $httpBackend.expectGET('/api/search');
    });

})
```

Espiando en un servicio angular que no llama al servicio de back-end

```
function calculatorService() {
    const service = {};
    service.add = function(a,b) {
        return a + b
    }

    return service;
}
```

```
angular.module('app').factory('calculatorService', calculatorService);
```

Pruebas

```
describe('calculator service', function() {
    var calculatorService;
    beforeEach(angular.mock.module('app'));

    beforeEach(inject(function(_calculatorService_) {
         calculatorService = _calculatorService_;
    }));

    it('should should add two numbers', function(){
        var actual = calculatorService.add(1,2);
        expect(actual).toBe(3);
    });
})
```

Espiando una propiedad

```
const foop = {
    get value() {},
    set value(v) {}
};

it('can spy on getter', () => {
    spyOnProperty(foop, 'value', 'get').and.returnValue(1);
    expect(foop.value).toBe(1);
});

it('and on setters', () => {
    const spiez = spyOnProperty(foop, 'value', 'set');
    foop.value = true;
    expect(spiez).toHaveBeenCalled();
});
```

Lea Espías en línea: https://riptutorial.com/es/jasmine/topic/1979/espias

Capítulo 4: Matchers personalizados

Examples

Añadiendo Matchers personalizados

Los emparejadores personalizados se pueden agregar en jasmine usando la sintaxis:

```
jasmine.addMatchers([
    toMatch: function () {
    return {
        compare: function (actual, expected) {
            return {
                pass: actual===expected,
                message: "Expected actual to match expected
            }
        }
    }
}
]);
```

Este emparejador ahora puede ser llamado con:

```
expected(actual).toMatch(expected);
```

Matchers Negativos

Los emparejadores personalizados tendrán su valor de paso negado cuando usen 'no'. Los emparejadores personalizados pueden tener un atributo de comparación negativo para manejar explícitamente los casos en que se desea su negación:

```
toMatch: function () {
        return {
            compare: function (actual, expected) {
                return {
                    pass: actual===expected,
                    message: "Expected actual to match expected"
                }
            },
            negativeCompare: function(actual, expected){
                return {
                    pass: actual!==expected,
                    message: "Expected actual not to match expected"
                }
            }
        }
    }
```

Lea Matchers personalizados en línea: https://riptutorial.com/es/jasmine/topic/6945/matchers-personalizados

Creditos

S. No	Capítulos	Contributors
1	Empezando con el jazmín	Akxe, cezar, Community, dev ツ, Matas Vaitkevicius, Paul Sweatte, Sibeesh Venu, tkwargs
2	Atributos	Sibeesh Venu
3	Espías	Ahmed Ahmed, Ben McCormick, gmuraleekrishna, hansmaad, Laoujin, Paul Sweatte
4	Matchers personalizados	SilentLupin