Framework Open Source para realizar unit testing en Front End

Carlos Cabrera - Arturo Garcia Reinhold - Maximiliano Guerra

Jasmine

Jasmine es un framework de BDD (Behavior Driven Development), es decir, un desarrollo orientado al comportamiento para JavaScript. No depende de ninguna otra librería. No requiere un DOM.

Jasmine nos proporciona las herramientas más básicas que necesitamos para probar nuestro código.

Básicamente se pueden utilizar las siguientes:

-describe(a, b): en a ponemos la descripción de un conjunto de tests, b sería una función anónima que se ejecuta con todos los tests incluidos.

-it(a,b): a sería la descripción de lo que se prueba, para tener una referencia, y b en este caso también sería una función anónima.

-expect(a): a es el valor a evaluar. Ejemplo: expect(a).not.toBe(false) // si esperamos un argumento verdadero.

-beforeAll(a): donde “a” será la función que se ejecutará antes de iniciar las pruebas.
-afterAll(a): donde “a” será la función que se ejecutará después de iniciar las pruebas.
-beforeEach(a): donde “a” será la función que se ejecutará antes de cada prueba.
-afterEach(a): donde “a” será la función que se ejecutará después de cada prueba.

Ejemplo de código para testear en una calculadora que incluye algunos métodos básicos (suma, resta, multiplicación y división):

```
var calc = {
    add: function (a, b) {
      return a + b;
    },
    substract: function (a, b) {
      return a - b;
    }
    //some functions ommited
};
```

Lo que se busca es que la calculadora funcione correctamente, que sume, reste, multiplique y divida bien.
A continuación una serie de test de ejemplos:

```
describe("Calc test suite:", function() { //comienza el test
   //suma de 2 enteros
   it("should add two numbers", function () { //en el describe se sitúan los
comentarios
    var a = 12,
        b = 5;
    var res = calc.add(a, b);
    expect(res).toBe(17);
   });
```

```
it("should sustract second number from the first", function () {//resta
    var a = 12,
        b = 5;
    var res = calc.substract(a, b);
    expect(res).toBe(7);
    expect(res).not.toBe(17);
  });
  it("should divide first number by the second", function () {/división
    var a = 10,
        b = 5;
    var res = calc.divide(a, b);
    expect(res).toBe(2);
  });
});//ommited multiply test
```

Jasmine

describe()
Punto de partida a todos los test. Se indica al principio y sirve para agrupar tests.

it()
Sirve para enumerar las distintas pruebas que se hacen.
it(“Debe sumar 2 números”…
it(“Debe restar 2 números”…
it(“Debe multiplicar 2 números”…
it(“Debe dividir 2 números”…

expect()
Sirve para cuando se obtiene el resultado obtenido poder aplicarle comparaciones (matchers) y comprobar lo que se espera.

- expect(x).toEqual(y);
- expect(x).toBeDefined();
- expect(x).toBeUndefined();
- expect(x).toBeNull();
- expect(x).toBeTruthy();
- expect(x).toBeFalsy();
- expect(x).toContain(y);
- expect(x).toBeLessThan(y);
- expect(x).toBeGreaterThan(y);
- expect(x).toMatch(pattern);
- expect(x).toBe(y);

Jasmine

Algunos ejemplos de expect:

```
describe('Hello world', function () {
    it('says world', function () {
        expect(helloWorld()).toContain('world');
    });
});
```

```
describe('Hello world', function () {
  it('says hello', function () {
    expect(helloWorld()).toEqual('Hello world!');
  });
});
```

Jasmine

Ejemplo de expect(matches) propios:

En este caso se realizó un expect propio con un nombre creado por el usuario que puntualmente controla que un número sea divisible por 2.

```
describe('Hello world', function () {

  beforeEach(function () {
    this.addMatchers({
      toBeDivisibleByTwo: function () {
        return (this.actual % 2) === 0;
      }
    });
  });

  it('is divisible by 2', function () {
    expect(gimmeANumber()).toBeDivisibleByTwo();
  });

});
```