**JASMINE FORO**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**beforeEach /**

Un beforeEach para varios it

beforeEach(function() {

localStorage.clear();

});

describe('My tests', function() {

describe('Test localstorage', function() {

it('Adds an item to localStorage', function() {

localStorage.setItem('foo', 'bar');

expect(localStorage.getItem('foo')).toBe('bar');

});

it('Is now empty because our beforeEach cleared localStorage', function() {

expect(localStorage.getItem('foo')).toBe(null);

});

});

});

////////////////////////////////////

A beforeEach block should not be inside of an it block.

/////////////////////////////////

TestBed.configureTestingModule needs to be called on every it statement

/////////////////////////////////

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*

**expect**

Arrojar un error

expect( function(){

parser.parse(raw);

} ).toThrow(new Error("Parsing is not possible"));

//////////////////////////////////////// /

Scaping ( in a regular expression

expect("test (test) with other stuff").toMatch("test \(test\)");

////////////////////////////////

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Fixture**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**inject**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Promise**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**toBeTruthy /**

toBe()

expect().toBe() is defined as:

function toBe() {

return {

compare: function(actual, expected) {

return {

pass: actual === expected

};

}

};

}

expect(foo).toBe(true), it will pass only if foo actually has the value true. Truthy values won't make the test pass.

toBeTruthy()

expect().toBeTruthy() is defined as:

function toBeTruthy() {

return {

compare: function(actual) {

return {

pass: !!actual

};

}

};

}

////////////////////////////////////////// /

Type coercion

A value is truthy if the coercion of this value to a boolean yields the value true.

The operation !! tests for truthiness by coercing the value passed to expect to a boolean.

Contrarily to what the currently accepted answer implies, == true is not a correct test for truthiness. You'll get funny things like

> "hello" == true

false

> "" == true

false

> [] == true

false

> [1, 2, 3] == true

False

Whereas using !! yields:

> !!"hello"

true

> !!""

false

> !![1, 2, 3]

true

> !![]

true

(Yes, empty or not, an array is truthy (veraz))

//////////////////////////////////////////////////////// /

toBeTrue()

expect().toBeTrue() is part of Jasmine-Matchers (which is registered on npm as jasmine-expect after a later project registered jasmine-matchers first).

expect().toBeTrue() is defined as:

function toBeTrue(actual) {

return actual === true ||

is(actual, 'Boolean') &&

actual.valueOf();

}

The difference with expect().toBeTrue() and expect().toBe(true) is that expect().toBeTrue() tests whether it is dealing with a Boolean object. expect(new Boolean(true)).toBe(true) would fail whereas expect(new Boolean(true)).toBeTrue() would pass. This is because of this funny thing:

> new Boolean(true) === true

false

> new Boolean(true) === false

False

At least it is truthy:

> !!new Boolean(true)

true

///////////////////////////////////// /

Check if a function has been called with a truthy value

let mySpy = spyOn(myComponent, 'myFunctionToSpyOn').and.callThrough();

// do something that triggers the function call

let arg = mySpy.calls.mostRecent().args[0];

expect(arg).toBeTruthy();

////////////////////////////////////