



This repository Search

Explore Gist Blog Help



OliPelz



LearnBoost / expect.js

Watch 42

Star 842

Fork 130

Minimalistic BDD-style assertions for Node.JS and the browser.

72 commits

1 branch

6 releases

18 contributors



branch: master

expect.js / +



README: update

guille authored on 20 Feb

latest commit 564e155e60



support	Initial release.	3 years ago
test	Add .withArgs() syntax for building scenario	a year ago
.gitignore	Initial release.	3 years ago
.npmignore	Initial release.	3 years ago
History.md	Release 0.3.0	9 months ago
Makefile	fix 'make test' for recent mochas	2 years ago
README.md	README: update	9 months ago
index.js	index: bump version	9 months ago
package.json	Release 0.3.1	9 months ago

README.md

Expect

Minimalistic BDD assertion toolkit based on [should.js](#)

```
expect(window.r).to.be(undefined);
expect({ a: 'b' }).to.eql({ a: 'b' })
expect(5).to.be.a('number');
expect([]).to.be.an('array');
expect(window).not.to.be.an(Image);
```

Features

- Cross-browser: works on IE6+, Firefox, Safari, Chrome, Opera.
- Compatible with all test frameworks.
- Node.JS ready (`require('expect.js')`).
- Standalone. Single global with no prototype extensions or shims.

How to use

Node

Install it with NPM or add it to your `package.json` :

```
$ npm install expect.js
```

Then:

```
var expect = require('expect.js');
```



Code



Issues

42



Pull Requests

27



Wiki



Pulse



Graphs

HTTPS clone URL

<https://github.com/>



You can clone with [HTTPS](#), [SSH](#), or [Subversion](#).

Clone in Desktop

Download ZIP

Browser

Expose the `index.js` found at the top level of this repository.

```
<script src="expect.js"></script>
```

API

ok: asserts that the value is *truthy* or not

```
expect(1).to.be.ok();
expect(true).to.be.ok();
expect({}).to.be.ok();
expect(0).to.not.be.ok();
```

be / equal: asserts `===` equality

```
expect(1).to.be(1)
expect(NaN).not.to.equal(NaN);
expect(1).not.to.be(true)
expect('1').to.not.be(1);
```

eq: asserts loose equality that works with objects

```
expect({ a: 'b' }).to.eq({ a: 'b' });
expect(1).to.eq('1');
```

a/an: asserts `typeof` with support for `array` type and `instanceof`

```
// typeof with optional `array`
expect(5).to.be.a('number');
expect([]).to.be.an('array'); // works
expect([]).to.be.an('object'); // works too, since it uses `typeof`

// constructors
expect(5).to.be.a(Number);
expect([]).to.be.an(Array);
expect(tobi).to.be.a(Ferret);
expect(person).to.be.a(Mammal);
```

match: asserts `String` regular expression match

```
expect(program.version).to.match(/[0-9]+\.[0-9]+\.[0-9]+/);
```

contain: asserts `indexOf` for an array or string

```
expect([1, 2]).to.contain(1);
expect('hello world').to.contain('world');
```

length: asserts `array.length`

```
expect([]).to.have.length(0);
expect([1, 2, 3]).to.have.length(3);
```

empty: asserts that an array is empty or not

```
expect([]).to.be.empty();
expect({}).to.be.empty();
expect({ length: 0, duck: 'typing' }).to.be.empty();
```

```
expect({ my: 'object' }).to.not.be.empty();
expect([1,2,3]).to.not.be.empty();
```

property: asserts presence of an own property (and value optionally)

```
expect(window).to.have.property('expect')
expect(window).to.have.property('expect', expect)
expect({a: 'b'}).to.have.property('a');
```

key/keys: asserts the presence of a key. Supports the `only` modifier

```
expect({ a: 'b' }).to.have.key('a');
expect({ a: 'b', c: 'd' }).to.only.have.keys('a', 'c');
expect({ a: 'b', c: 'd' }).to.only.have.keys(['a', 'c']);
expect({ a: 'b', c: 'd' }).to.not.only.have.key('a');
```

throwException/throwError: asserts that the `Function` throws or not when called

```
expect(fn).toThrowError(); // synonym of throwException
expect(fn).toThrowError(function (e) { // get the exception object
  expect(e).to.be.a(SyntaxError);
});
expect(fn).toThrowError(/matches the exception message/);
expect(fn2).to.not.throwException();
```

withArgs: creates anonymous function to call `fn` with arguments

```
expect(fn).withArgs(invalid, arg).toThrowException();
expect(fn).withArgs(valid, arg).to.not.throwException();
```

within: asserts a number within a range

```
expect(1).to.be.within(0, Infinity);
```

greaterThan/above: asserts `>`

```
expect(3).to.be.above(0);
expect(5).to.be.greaterThan(3);
```

lessThan/below: asserts `<`

```
expect(0).to.be.below(3);
expect(1).to.be.lessThan(3);
```

fail: explicitly forces failure.

```
expect().fail()
expect().fail("Custom failure message")
```

Using with a test framework

For example, if you create a test suite with [mocha](#).

Let's say we wanted to test the following program:

math.js

```
function add (a, b) { return a + b; };
```

Our test file would look like this:

```
describe('test suite', function () {
  it('should expose a function', function () {
    expect(add).to.be.a('function');
  });

  it('should do math', function () {
    expect(add(1, 3)).to.equal(4);
  });
});
```

If a certain expectation fails, an exception will be raised which gets captured and shown/processed by the test runner.

Differences with should.js

- No need for static `should` methods like `should.strictEqual` . For example, `expect(obj).to.be(undefined)` works well.
- Some API simplifications / changes.
- API changes related to browser compatibility.

Running tests

Clone the repository and install the developer dependencies:

```
git clone git://github.com/LearnBoost/expect.js.git expect
cd expect && npm install
```

Node

```
make test
```

Browser

```
make test-browser
```

and point your browser(s) to `http://localhost:3000/test/`

Credits

(The MIT License)

Copyright (c) 2011 Guillermo Rauch <guillermo@learnboost.com>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the 'Software'), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

3rd-party

Heavily borrows from [should.js](#) by TJ Holowaychuk - MIT.

