Introduction to JavaScript unit testing

Vladimir Alekseichenko

Agenda

- Node Packaged Modules
- Jasmine
- Mocha + (Should & Chai)
- QUnit
- Buster
- Karma
- Testem

NodeJS
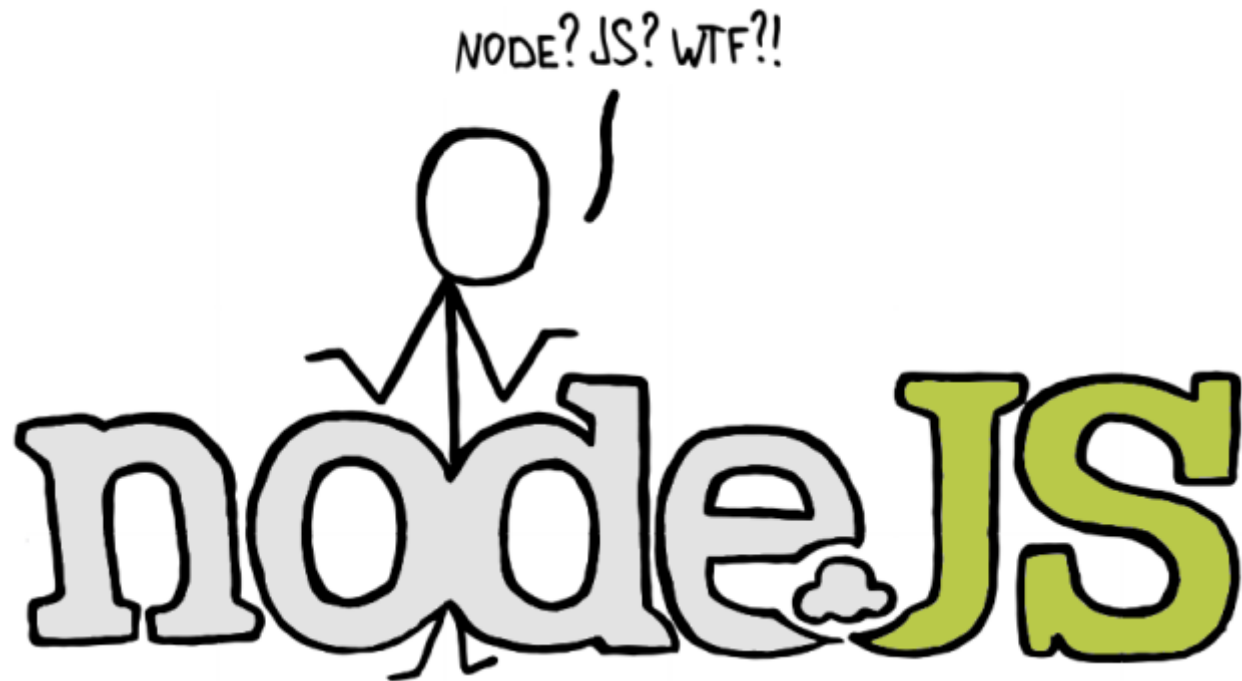
Install nodeJS

nodejs.org/download

Node Packaged Modules

It is the official package manager for Node.js

npm

- npm **init**

- npm **install** <name>

- npm **search** [search terms …]

- npm **run-script** <script> <name>

- …

package.json

Specifics of npm's **package.json** handling

- name

- version

- description

- scripts

- author

- dependencies

- …

Dependencies (package.json)

Dependencies are specified with a simple hash of package name to version range.

- version - *Must match version exactly.*

- >version - *Must be greater than version.*

- >=version

- ~version - *Approximately equivalent to version.*

- *…*

Dependencies (package.json)

```json
{ "dependencies" :
  { "foo" : "1.0.0 - 2.9999.9999"
  , "bar" : ">=1.0.2 <2.1.2"
  , "baz" : ">1.0.2 <=2.3.4"
  , "boo" : "2.0.1"
  , "qux" : "<1.0.0 || >=2.3.1 <2.4.5 || >=2.5.2 <3.0.0"
  , "asd" : "http://asdf.com/asdf.tar.gz"
  , "til" : "~1.2"
  , "elf" : "~1.2.3"
  , "two" : "2.x"
  , "thr" : "3.3.x"
  }
}
```

npm **init**

Interactively create a **package.json** file

This will ask you a bunch of questions, and then write a package.json for you.

npm **install** <name>

Install a package

- npm **install** (with no args in a package dir)
- npm **install** <tarball file>
- npm **install** <tarball url>
- ...

npm **install** == npm **i**

npm **install**
useful options

- npm **install** <name> **--save**

- npm **install** <name> **--save-dev**

- npm **install** <name> **--production**

npm **search** [search terms …]

Search the registry for packages matching the search terms.

- npm **search** [search terms …]
- npm **s** [search terms …]
- npm **se** [search terms …]

npm **run-script** <script> <name>

This runs an arbitrary command from a package's "scripts" object.

- npm **run-script** <script> <name>

Task 1

Jasmine

 is a

behavior-**d**riven **d**evelopment framework for

testing **JavaScript** code.

Jasmine

- It does not depend on any other JavaScript frameworks.

- It does not require a DOM.

- And it has a clean, obvious syntax so that you can easily write tests.

Installation (1)

`npm install `**`jasmine-node`**

`npm i `**`jasmine-node`**` --save-dev`

Installation (2)

Download jasmine-standalone-x.x.x.zip from
https://github.com/pivotal/jasmine/downloads:

- **lib**
  - **jasmine-x.x.x**
    - **jasmine.css**
    - **jasmine.js**
    - **jasmine-html.js**
- spec
  - PlaySpec.js
  - SpecHelper.js
- src
  - Player.js
  - Song.js
- **SpecRunner.html**

Installation (3)

```
PM> Install-Package jasmine-js
```

- **Scripts**
  - jasmine-html.js
  - jasmine.js

Report

```
Jasmine 1.3.0 revision 1354052693

• • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • • •

Passing 33 specs

Test suite
  adds two numbers together

Learning the matchers
  compares
    compares using ===
    compares variables and objects (including content)

  defined or not
    checks value to be defined
    checks value to be undefined

  to be null
    checks value to be null

  truthy or falsy
    checks value to be true
    checks value to be false

  less or greater
    is less than 10
    is greater than 10
    checks value to be close to

  match
```

First test

```javascript
describe("Test suite", function() {
    it("adds two numbers together", function() {
        expect(1 + 2).toBe(3);
    });
});

describe("Disabled", function() {
    xdescribe("disabled suite", function() {
        it("will not run, since the suite has been disabled", function() {
            expect(true).toBe(true);
        });
    });

    xit("disabled test", function() {
        expect(true).toBe(true);
    });
});
```

First test – describe()

```
describe("Test suite", function() {
    it("adds two numbers together", function() {
        expect(1 + 2).toBe(3);
    });
});

describe("Disabled", function() {
    xdescribe("disabled suite", function() {
        it("will not run, since the suite has been disabled", function() {
            expect(true).toBe(true);
        });
    });

    xit("disabled test", function() {
        expect(true).toBe(true);
    });
});
```

First test – it()

```
describe("Test suite", function() {
    it("adds two numbers together", function() {
        expect(1 + 2).toBe(3);
    });
});

describe("Disabled", function() {
    xdescribe("disabled suite", function() {
        it("will not run, since the suite has been disabled", function() {
            expect(true).toBe(true);
        });
    });

    xit("disabled test", function() {
        expect(true).toBe(true);
    });
});
```

First test – xdescribe()

```javascript
describe("Test suite", function() {
    it("adds two numbers together", function() {
        expect(1 + 2).toBe(3);
    });
});

describe("Disabled", function() {
    xdescribe("disabled suite", function() {
        it("will not run, since the suite has been disabled", function() {
            expect(true).toBe(true);
        });
    });

    xit("disabled test", function() {
        expect(true).toBe(true);
    });
});
```

First test – xit()

```
describe("Test suite", function() {
    it("adds two numbers together", function() {
        expect(1 + 2).toBe(3);
    });
});

describe("Disabled", function() {
    xdescribe("disabled suite", function() {
        it("will not run, since the suite has been disabled", function() {
            expect(true).toBe(true);
        });
    });

    xit("disabled test", function() {
        expect(true).toBe(true);
    });
});
```

```javascript
describe('defined or not', function() {
    it("checks value to be defined", function() {
        expect(window.document).toBeDefined();
    });

    it("checks value to be undefined", function() {
        expect(window.notExists).toBeUndefined();
    });
});
```

```javascript
describe('defined or not', function() {
    it("checks value to be defined", function() {
        expect(window.document).toBeDefined();
    });

    it("checks value to be undefined", function() {
        expect(window.notExists).toBeUndefined();
    });
});
```

```javascript
describe('defined or not', function() {
    it("checks value to be defined", function() {
        expect(window.document).toBeDefined();
    });

    it("checks value to be undefined", function() {
        expect(window.notExists).toBeUndefined();
    });
});
```

```javascript
describe('to be null', function() {
    it("checks value to be null", function() {
        var a;
        a = null;
        return expect(a).toBeNull();
    });
});

describe('truthy or falsy', function() {
    it("checks value to be true", function() {
        expect(5 > 0).toBeTruthy();
    });

    it("checks value to be false", function() {
        expect(5 < 0).toBeFalsy();
    });
});
```

```javascript
describe('to be null', function() {
    it("checks value to be null", function() {
        var a;
        a = null;
        return expect(a).toBeNull();
    });
});

describe('truthy or falsy', function() {
    it("checks value to be true", function() {
        expect(5 > 0).toBeTruthy();
    });

    it("checks value to be false", function() {
        expect(5 < 0).toBeFalsy();
    });
});
```

```javascript
describe('to be null', function() {
    it("checks value to be null", function() {
        var a;
        a = null;
        return expect(a).toBeNull();
    });
});

describe('truthy or falsy', function() {
    it("checks value to be true", function() {
        expect(5 > 0).toBeTruthy();
    });

    it("checks value to be false", function() {
        expect(5 < 0).toBeFalsy();
    });
});
```

```javascript
describe('to be null', function() {
    it("checks value to be null", function() {
        var a;
        a = null;
        return expect(a).toBeNull();
    });
});

describe('truthy or falsy', function() {
    it("checks value to be true", function() {
        expect(5 > 0).toBeTruthy();
    });

    it("checks value to be false", function() {
        expect(5 < 0).toBeFalsy();
    });
});
```

```javascript
describe('match', function() {
    it("outputs the right text", function () {
        expect("123.34").toMatch(/\d+\.\d{2}/);
        expect("123.34").not.toMatch(/string/);
    });
});

describe('to contain', function() {
    it("should contain oranges", function () {
        expect([1, 2, 3]).toContain(2);
        expect("one two three string").toContain("two");
    });
});
```

```javascript
describe('match', function() {
    it("outputs the right text", function () {
        expect("123.34").toMatch(/\d+\.\d{2}/);
        expect("123.34").not.toMatch(/string/);
    });
});

describe('to contain', function() {
    it("should contain oranges", function () {
        expect([1, 2, 3]).toContain(2);
        expect("one two three string").toContain("two");
    });
});
```

```javascript
describe('match', function() {
    it("outputs the right text", function () {
        expect("123.34").toMatch(/\d+\.\d{2}/);
        expect("123.34").not.toMatch(/string/);
    });
});

describe('to contain', function() {
    it("should contain oranges", function () {
        expect([1, 2, 3]).toContain(2);
        expect("one two three string").toContain("two");
    });
});
```

```javascript
describe('exception', function() {
    it("throws exception", function() {
        var func = function() {
            window.notExists.value;
        };

        expect(func).toThrow();
    });
});
```

```javascript
describe('exception', function() {
    it("throws exception", function() {
        var func = function() {
            window.notExists.value;
        };

        expect(func).toThrow();
    });
});
```

```javascript
describe('exception', function() {
    it("throws exception", function() {
        var func = function() {
            window.notExists.value;
        };

        expect(func).toThrow();
    });
});
```

```javascript
describe("A spec (with setup and tear-down)", function() {
    var foo;

    beforeEach(function() {
        foo = 0;
        foo += 1;
    });

    afterEach(function() {
        foo = 0;
    });

    it("is just a function, so it can contain any code", function() {
        expect(foo).toEqual(1);
    });

    it("can have more than one expectation", function() {
        expect(foo).toEqual(1);
    });
});
```

```javascript
describe("A spec (with setup and tear-down)", function() {
    var foo;

    beforeEach(function() {
        foo = 0;
        foo += 1;
    });

    afterEach(function() {
        foo = 0;
    });

    it("is just a function, so it can contain any code", function() {
        expect(foo).toEqual(1);
    });

    it("can have more than one expectation", function() {
        expect(foo).toEqual(1);
    });
});
```

```javascript
describe("A spec (with setup and tear-down)", function() {
    var foo;

    beforeEach(function() {
        foo = 0;
        foo += 1;
    });

    afterEach(function() {
        foo = 0;
    });

    it("is just a function, so it can contain any code", function() {
        expect(foo).toEqual(1);
    });

    it("can have more than one expectation", function() {
        expect(foo).toEqual(1);
    });
});
```

```javascript
describe("A spec (with setup and tear-down)", function() {
    var foo;

    beforeEach(function() {
        foo = 0;
        foo += 1;
    });

    afterEach(function() {
        foo = 0;
    });

    it("is just a function, so it can contain any code", function() {
        expect(foo).toEqual(1);
    });

    it("can have more than one expectation", function() {
        expect(foo).toEqual(1);
    });
});
```

```javascript
describe("Asynchronous", function() {
    var a = 0;

    it("async executes code", function() {

        runs(function() {
            setTimeout(function() {
            a = 5;
          }, 100);
        });

        waitsFor(function() {
          return a === 5;
        }, "the value should be changed", 150);
    });
});
```

```javascript
describe("Asynchronous", function() {
    var a = 0;

    it("async executes code", function() {

        runs(function() {
            setTimeout(function() {
            a = 5;
          }, 100);
        });

        waitsFor(function() {
            return a === 5;
        }, "the value should be changed", 150);
    });
});
```

```javascript
describe("Asynchronous", function() {
    var a = 0;

    it("async executes code", function() {

        runs(function() {
            setTimeout(function() {
                a = 5;
            }, 100);
        });

        waitsFor(function() {
            return a === 5;
        }, "the value should be changed", 150);
    });
});
```

```javascript
describe("Asynchronous", function() {
    var a = 0;

    it("async executes code", function() {

        runs(function() {
            setTimeout(function() {
            a = 5;
        }, 100);
        });

        waitsFor(function() {            optional timeout
            return a === 5;
        }, "the value should be changed", 150);
    });
});
```

1
2

```javascript
describe('Writing custom matchers', function() {
    beforeEach(function () {
        this.addMatchers({
            toBeBetween: function (rangeFloor, rangeCeiling) {
                if (rangeFloor > rangeCeiling) {
                    var temp = rangeFloor;
                    rangeFloor = rangeCeiling;
                    rangeCeiling = temp;
                }
                return this.actual > rangeFloor && this.actual < rangeCeiling;
            }
        });
    });

    it("is between 5 and 30", function () {
        expect(10).toBeBetween(5, 30);
    });
    it("is between 30 and 500", function () {
        expect(100).toBeBetween(500, 30);
    });
});
```

```javascript
describe('Writing custom matchers', function() {
    beforeEach(function () {
        this.addMatchers({
            toBeBetween: function (rangeFloor, rangeCeiling) {
                if (rangeFloor > rangeCeiling) {
                    var temp = rangeFloor;
                    rangeFloor = rangeCeiling;
                    rangeCeiling = temp;
                }
                return this.actual > rangeFloor && this.actual < rangeCeiling;
            }
        });
    });

    it("is between 5 and 30", function () {
        expect(10).toBeBetween(5, 30);
    });
    it("is between 30 and 500", function () {
        expect(100).toBeBetween(500, 30);
    });
});
```

```javascript
describe('Writing custom matchers', function() {
    beforeEach(function () {                                    SpecHelper.js
        this.addMatchers({
            toBeBetween: function (rangeFloor, rangeCeiling) {
                if (rangeFloor > rangeCeiling) {
                    var temp = rangeFloor;
                    rangeFloor = rangeCeiling;
                    rangeCeiling = temp;
                }
                return this.actual > rangeFloor && this.actual < rangeCeiling;
            }
        });
    });

    it("is between 5 and 30", function () {
        expect(10).toBeBetween(5, 30);
    });
    it("is between 30 and 500", function () {
        expect(100).toBeBetween(500, 30);
    });
});
```
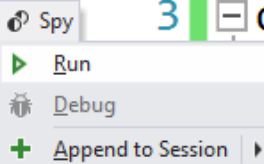
```javascript
describe('Writing custom matchers', function() {
    beforeEach(function () {
        this.addMatchers({
            toBeBetween: function (rangeFloor, rangeCeiling) {
                if (rangeFloor > rangeCeiling) {
                    var temp = rangeFloor;
                    rangeFloor = rangeCeiling;
                    rangeCeiling = temp;
                }
                return this.actual > rangeFloor && this.actual < rangeCeiling;
            }
        });
    });

    it("is between 5 and 30", function () {
        expect(10).toBeBetween(5, 30);
    });
    it("is between 30 and 500", function () {
        expect(100).toBeBetween(500, 30);
    });
});
```

Run jasmine in R# 7

Run jasmine in R# 7

```
1   /// <reference path="~/lib/jasmine-1.3.0/jasmine.js"/>
2   /// <reference path="~/src/Person.js"/>
3   describe("Spy", function () {
        var person = null, expectedValue = null;
        beforeEach(function() {
6           person = new Person("Jim", 25);
7           expectedValue = "Julia";
8       });
9
```
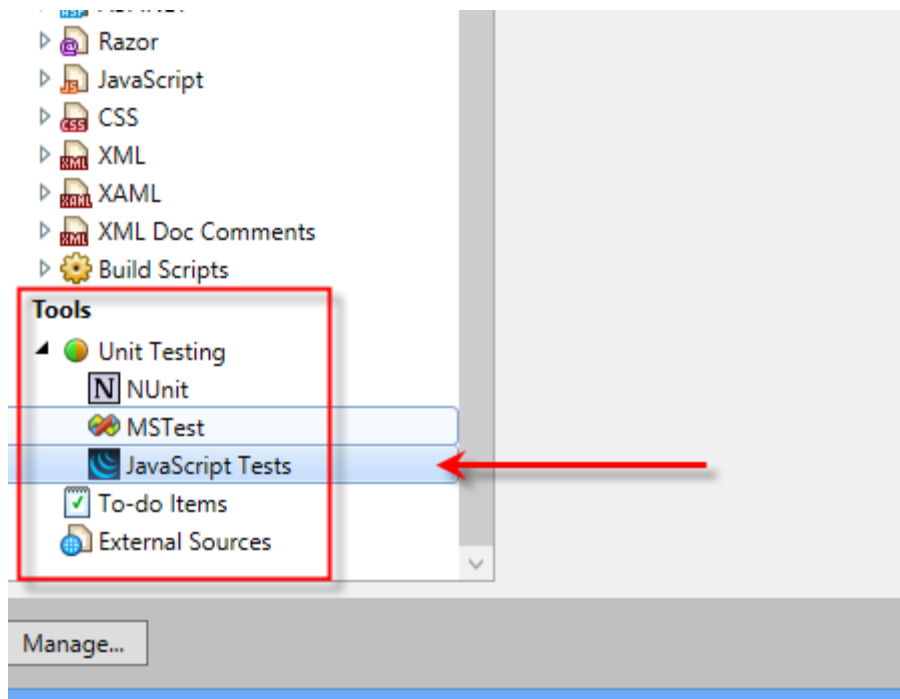
Spy
▷ Run
🐞 Debug
➕ Append to Session ▷

Run jasmine in R# 7

```
1   /// <reference path="~/lib/jasmine-1.3.0/jasmine.js"/>
2   /// <reference path="~/src/Person.js"/>
3   describe("Spy", function () {
        var person = null, expectedValue = null;
        beforeEach(function() {
6           person = new Person("Jim", 25);
7           expectedValue = "Julia";
8       });
9
```

Spy
▷ Run
🐞 Debug
➕ Append to Session ▶

Run jasmine in R# 7

```
1    /// <reference path="~/lib/jasmine-1.3.0/jasmine.js"/>
2    /// <reference path="~/src/Person.js"/>
3  ⊟ describe("Spy", function () {
       var person = null, expectedValue = null;
       beforeEach(function() {
6          person = new Person("Jim", 25);
7          expectedValue = "Julia";
8      });
9
```

```
🐞 Spy
▷ Run
🐞 Debug
➕ Append to Session ▶
```

Run jasmine in R# 7

PhantomJs + Jasmine + R# 7

PhantomJs + Jasmine + R# 7

JavaScript Tests

Supported Frameworks

☑ Enable QUnit support

☑ Enable Jasmine support

Run Tests With

○ Browser

Select browser   System Default Browser

◉ PhantomJS

Path to PhantomJS executable   D:\bin\phantomjs-1.9.2\phantomjs.exe   Browse...

PhantomJs + Jasmine + R# 7

Task 2

mocha

simple, flexible, fun

Mocha

It's a feature-rich JavaScript test framework running on node.js and the browser, making asynchronous testing simple and fun.

Features

- simple async suport
- test coverage reporting
- proper exit status for CI support etc
- auto-detects and disables coloring for non-ttys
- file watcher suport
- global variable leak detection
- ...

Installation

```
npm install mocha
```

```
npm i mocha --save-dev
```

mocha --help

```
Usage: mocha [debug] [options] [files]

Commands:

  init <path>
  initialize a client-side mocha setup at <path>

Options:

  -h, --help                      output usage information
  -V, --version                   output the version number
  -r, --require <name>            require the given module
  -R, --reporter <name>           specify the reporter to use
  -u, --ui <name>                 specify user-interface (bdd|tdd|exports)
  -g, --grep <pattern>            only run tests matching <pattern>
  -i, --invert                    inverts --grep matches
  -t, --timeout <ms>              set test-case timeout in milliseconds [2000]
  -s, --slow <ms>                 "slow" test threshold in milliseconds [75]
  -w, --watch                     watch files for changes
  -c, --colors                    force enabling of colors
  -C, --no-colors                 force disabling of colors
  -G, --growl                     enable growl notification support
  -d, --debug                     enable node's debugger, synonym for node --debug
  -b, --bail                      bail after first test failure
  -A, --async-only                force all tests to take a callback (async)
  -S, --sort                      sort test files
  --recursive                     include sub directories
  --debug-brk                     enable node's debugger breaking on the first line
  --globals <names>               allow the given comma-delimited global [names]
  --check-leaks                   check for global variable leaks
  --interfaces                    display available interfaces
  --reporters                     display available reporters
  --compilers <ext>:<module>,...  use the given module(s) to compile files
```

JavaScript assertion libraries

- Should (https://github.com/visionmedia/should.js)

- Chaijs (http://chaijs.com/)

- Expect (https://github.com/LearnBoost/expect.js)

- jShould (https://github.com/eliperelman/jShould)

- YUIPort (https://github.com/gso/YUIPort)

- …

Should

It's an expressive, readable, test framework agnostic, assertion library for node.

Installation

`npm install` **`should`**

Mocha + Should

```javascript
describe('truth', function () {
    it('should be true', function () {
        true.should.be.true;
    });

    it('should not be false', function () {
        true.should.not.be.false;
    });
});
```

```
describe('truth', function () {
    it('should be true', function () {
        true.should.be.true;
    });

    it('should not be false', function () {
        true.should.not.be.false;
    });
});
```

```javascript
describe('truth', function () {
    it('should be true', function () {
        true.should.be.true;
    });

    it('should not be false', function () {
        true.should.not.be.false;
    });
});
```

```javascript
it('equal & exactly', function() {
    (4).should.equal(4);
    'test'.should.equal('test');
    [1,2,3].should.not.equal([1,2,3]);
    (4).should.be.exactly(4);
});

it('within', function() {
    var age = 4;
    age.should.be.within(1, 100);
});

it('approximately', function() {
    (99.99).should.be.approximately(100, 0.1);
});

it('instanceof', function() {
    [].should.be.an.instanceof(Array);
    [].should.be.an.instanceOf(Array);
});
```

```
it('equal & exactly', function() {
    (4).should.equal(4);
    'test'.should.equal('test');
    [1,2,3].should.not.equal([1,2,3]);
    (4).should.be.exactly(4);
});

it('within', function() {
    var age = 4;
    age.should.be.within(1, 100);
});

it('approximately', function() {
    (99.99).should.be.approximately(100, 0.1);
});

it('instanceof', function() {
    [].should.be.an.instanceof(Array);
    [].should.be.an.instanceOf(Array);
});
```

```javascript
it('equal & exactly', function() {
    (4).should.equal(4);
    'test'.should.equal('test');
    [1,2,3].should.not.equal([1,2,3]);
    (4).should.be.exactly(4);
});

it('within', function() {
    var age = 4;
    age.should.be.within(1, 100);
});

it('approximately', function() {
    (99.99).should.be.approximately(100, 0.1);
});

it('instanceof', function() {
    [].should.be.an.instanceof(Array);
    [].should.be.an.instanceOf(Array);
});
```

```javascript
it('equal & exactly', function() {
    (4).should.equal(4);
    'test'.should.equal('test');
    [1,2,3].should.not.equal([1,2,3]);
    (4).should.be.exactly(4);
});

it('within', function() {
    var age = 4;
    age.should.be.within(1, 100);
});

it('approximately', function() {
    (99.99).should.be.approximately(100, 0.1);
});

it('instanceof', function() {
    [].should.be.an.instanceof(Array);
    [].should.be.an.instanceOf(Array);
});
```

```javascript
it('equal & exactly', function() {
    (4).should.equal(4);
    'test'.should.equal('test');
    [1,2,3].should.not.equal([1,2,3]);
    (4).should.be.exactly(4);
});

it('within', function() {
    var age = 4;
    age.should.be.within(1, 100);
});

it('approximately', function() {
    (99.99).should.be.approximately(100, 0.1);
});

it('instanceof', function() {
    [].should.be.an.instanceof(Array);
    [].should.be.an.instanceOf(Array);
});
```

```javascript
describe('throw', function() {
    it('assert an exception is thrown', function() {
        (function() {
            throw new Error('fail');
        }).should.throw();
    });

    it('assert an exception is not thrown', function() {
        (function() {}).should.not.throw();
    });

    it('assert exception message matches string', function() {
        (function() {
            throw new Error('fail');
        }).should.throw('fail');
    });

    it('throwError', function() {
        (function() {
            throw new Error('failed to baz');
        }).should.throwError(/^fail.*/);
    });
});
```
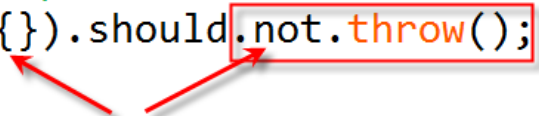
```javascript
describe('throw', function() {
    it('assert an exception is thrown', function() {
        (function() {
            throw new Error('fail');
        }).should.throw();
    });

    it('assert an exception is not thrown', function() {
        (function() {}).should.not.throw();
    });

    it('assert exception message matches string', function() {
        (function() {
            throw new Error('fail');
        }).should.throw('fail');
    });

    it('throwError', function() {
        (function() {
            throw new Error('failed to baz');
        }).should.throwError(/^fail.*/);
    });
});
```
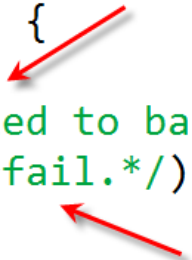
```
describe('throw', function() {
    it('assert an exception is thrown', function() {
        (function() {
            throw new Error('fail');
        }).should.throw();
    });

    it('assert an exception is not thrown', function() {
        (function() {}).should.not.throw();
    });

    it('assert exception message matches string', function() {
        (function() {
            throw new Error('fail');
        }).should.throw('fail');
    });

    it('throwError', function() {
        (function() {
            throw new Error('failed to baz');
        }).should.throwError(/^fail.*/);
    });
});
```

```javascript
describe('throw', function() {
    it('assert an exception is thrown', function() {
        (function() {
            throw new Error('fail');
        }).should.throw();
    });

    it('assert an exception is not thrown', function() {
        (function() {}).should.not.throw();
    });

    it('assert exception message matches string', function() {
        (function() {
            throw new Error('fail');
        }).should.throw('fail');
    });

    it('throwError', function() {
        (function() {
            throw new Error('failed to baz');
        }).should.throwError(/^fail.*/);
    });
});
```

```javascript
describe('throw', function() {
    it('assert an exception is thrown', function() {
        (function() {
            throw new Error('fail');
        }).should.throw();
    });

    it('assert an exception is not thrown', function() {
        (function() {}).should.not.throw();
    });

    it('assert exception message matches string', function() {
        (function() {
            throw new Error('fail');
        }).should.throw('fail');
    });

    it('throwError', function() {
        (function() {
            throw new Error('failed to baz');
        }).should.throwError(/^fail.*/);
    });
});
```

Chai

It is a BDD / TDD assertion library for node and the browser that can be delightfully paired with any JavaScript testing framework.

Installation

`npm install `**`chai`**

`npm i `**`chai`**` --save-dev`

Task 3

Photo by Vincent Laforet / The New York Times

http://www.netlore.ru/ty-dumaesh-u-tebya-ploxaya-rabota

QUnit

QUnit

It's a powerful, easy-to-use JavaScript unit testing framework.

It's used by the jQuery, jQuery UI and jQuery Mobile projects and is capable of testing any generic JavaScript code, including itself!

Installation (1)

```
npm install qunitjs
```

```
npm i qunitjs --save-dev
```

Installation (2)

Download two files from
http://codeorigin.jquery.com/qunit/:

- qunit-x.x.x.js
- qunit-x.x.x.css

Getting started

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>QUnit Example</title>
  <link rel="stylesheet" href="./lib/qunit/qunit-1.12.0.css">
</head>
<body>
  <div id="qunit"></div>
  <div id="qunit-fixture"></div>
  <script src="../src/Person.js"></script>
  <script src="../src/MyException.js"></script>

  <script src="./lib/qunit/qunit-1.12.0.js"></script>

  <script src="./Assert.js"></script>
  <script src="./AsyncControl.js"></script>
  <script src="./Callbacks.js"></script>
</body>
</html>
```

Getting started

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>QUnit Example</title>
  <link rel="stylesheet" href="./lib/qunit/qunit-1.12.0.css">
</head>
<body>
  <div id="qunit"></div>
  <div id="qunit-fixture"></div>
  <script src="../src/Person.js"></script>
  <script src="../src/MyException.js"></script>

  <script src="./lib/qunit/qunit-1.12.0.js"></script>

  <script src="./Assert.js"></script>
  <script src="./AsyncControl.js"></script>
  <script src="./Callbacks.js"></script>
</body>
</html>
```

Getting started

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>QUnit Example</title>
  <link rel="stylesheet" href="./lib/qunit/qunit-1.12.0.css">
</head>
<body>
  <div id="qunit"></div>
  <div id="qunit-fixture"></div>
  <script src="../src/Person.js"></script>
  <script src="../src/MyException.js"></script>

  <script src="./lib/qunit/qunit-1.12.0.js"></script>

  <script src="./Assert.js"></script>
  <script src="./AsyncControl.js"></script>
  <script src="./Callbacks.js"></script>
</body>
</html>
```

sources

tests

QUnit Example

Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/30.0.1599.101 Safari/537.36

Tests completed in 290 milliseconds.
33 assertions of 33 passed, 0 failed.

1. ok: ok (0, 2, 2) Rerun	2 ms
2. equal and notEqual: equal (0, 6, 6) Rerun	1 ms
3. equal and notEqual: scrictEqual (0, 2, 2) Rerun	1 ms
4. equal and notEqual: notStrictEqual (0, 4, 4) Rerun	1 ms
5. equal and notEqual: notEqual (0, 4, 4) Rerun	1 ms
6. deepEqual and notDeepEqual: deepEqual (0, 5, 5) Rerun	2 ms
7. deepEqual and notDeepEqual: notDeepEqual (0, 3, 3) Rerun	1 ms
8. throws exception: throw "error" (0, 3, 3) Rerun	1 ms
9. expect: test withc expect (0, 2, 2) Rerun	0 ms
10. asynchronous test: stop and start (0, 1, 1) Rerun	113 ms
11. asynchronous test: asyncTest - without explicitly stop() (0, 1, 1) Rerun	114 ms

QUnit Example

Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/30.0.1599.101 Safari/537.36

Tests completed in 363 milliseconds.
32 assertions of 33 passed, 1 failed.

1. ok: ok **(0, 2, 2)** Rerun   3 ms

2. **equal and notEqual: equal (1, 5, 6)** Rerun   6 ms

> 1. true equals true
>
> 2. 3 equals 3
> **Expected:** 3
>   **Result:** 2
>   **Diff:** 3 2
>   **Source:**   at Object.<anonymous>
>   (file:///D:/pr/intr_js_test/qunit/test/Assert.js:17:5)
>
> 3. empty string equals another empty string
>
> 4. Jim equals Jim
>
> 5. true
>
> 6. true

```
//module() - Group related tests under a single label.
module('ok');

//test() - Add a test to run.
//ok() - A boolean assertion, equivalent to CommonJS's
test("ok", function() {
    ok(true, 'true is ok');
    ok(person != null, 'Person is not null');
});
```

```javascript
module('equal and notEqual'); //equals -> equal
//A non-strict comparison assertion, roughly equivalent to JUnit assertEquals.
test('equal', function() {
    equal(true, true, 'true equals true');
    equal(2, 3, '3 equals 3');
    equal('', '', 'empty string equals another empty string');
    equal(person.getName(), 'Jim', 'Jim equals Jim');

    equal( 0, false, 'true');
    equal( null, undefined, 'true');
});

//strictEqual() - A strict type and value comparison assertion.
test('scrictEqual', function() {
    strictEqual(1, 1, '1 strictEqual 1');
    strictEqual('', '', 'empy string strictEqual empty string');

});
```

```
module('deepEqual and notDeepEqual'); //some -> deepEqual

//deepEqual() - A deep recursive comparison assertion, working on pri
test('deepEqual', function() {
    deepEqual([], [], '[] deepEqual []');
    deepEqual([1, 2, 3], [1, 2, 3], '[1, 2, 3] deepEqual [1, 2, 3]');
    deepEqual([[1], [2], [3]], [[1], [2], [3]], '[[1], [2], [3]] deep

    deepEqual({}, {}, '{} deepEqual {}');
    deepEqual(person, person, 'person deepEqual person');
});

//notDeepEqual() - An inverted deep recursive comparison assertion, w
test('notDeepEqual', function() {
    notDeepEqual([], [1, 2, 3], '[] notDeepEqual [1, 2, 3]');

    notDeepEqual( 0, false, '0 notDeepEqual false');
    notDeepEqual( null, undefined, 'null notDeepEqual undefined');
});
```

```javascript
module('throws exception');
//Assertion to test if a callback throws an exception when run.
test('throw "error"', function() {
    var myException = new MyException('some valuable message');
    throws(
        function() { throw 'error' },
        'throws with just a message, no expected'
    );

    throws(
        function() { throw myException; },
        MyException,
        "raised error is an instance of MyException"
    );

    throws(
        function() { throw myException; },
        /message/,
        'raised error message contains "message"'
    );
});
```
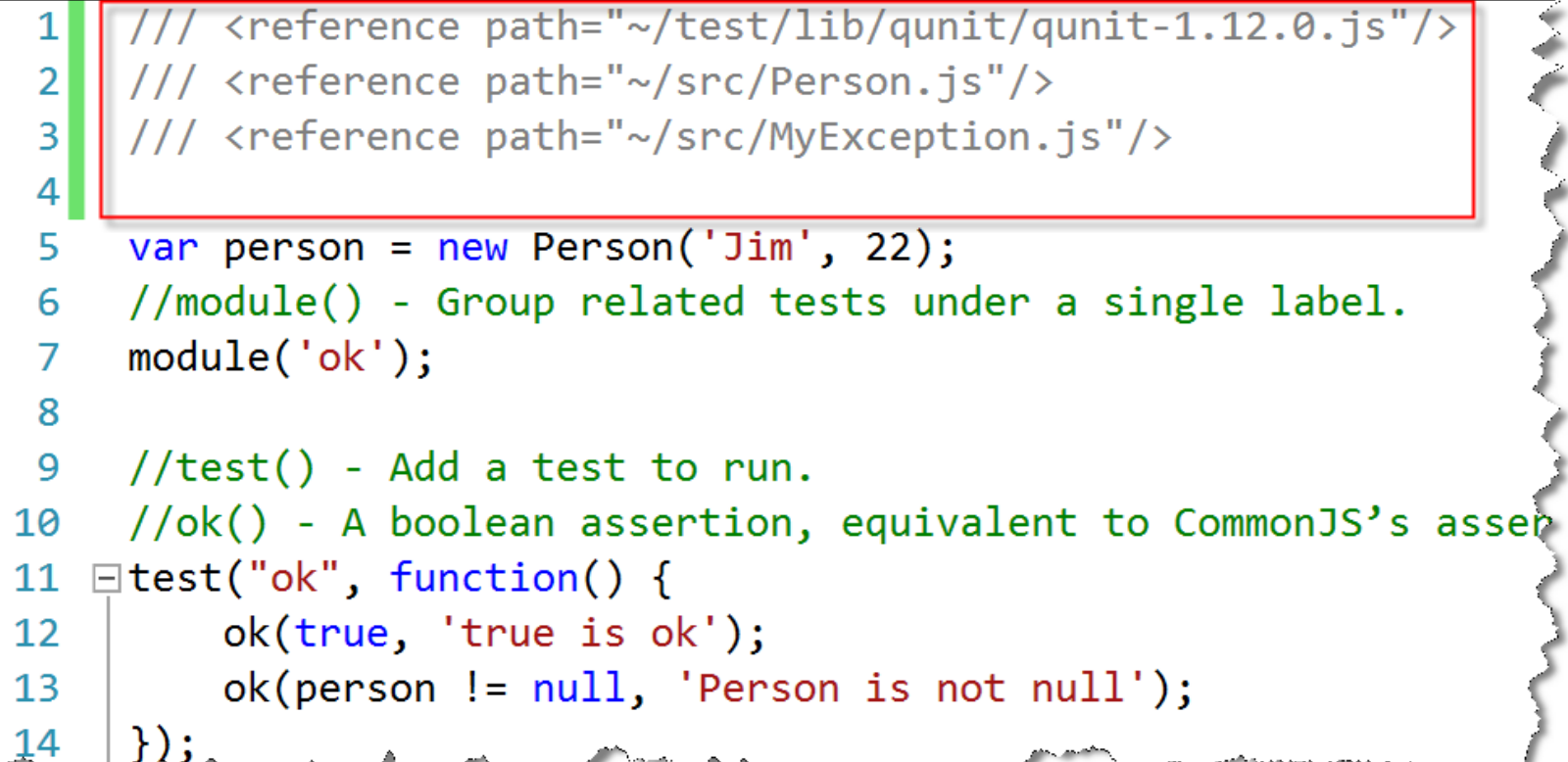
```javascript
test('stop and start', function() {
    // Pause the test first
    stop();

    setTimeout(function() {
        ok(true);

        // After the assertion has been called,
        // continue the test
        start();
    }, 100);
});
```

```javascript
asyncTest('asyncTest - without explicitly stop()', function() {

    setTimeout(function() {
        ok(true);

        // After the assertion has been called,
        // continue the test
        start();
    }, 100);
});
```

Run QUnit in R# 7
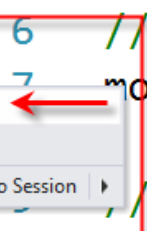
```
1   /// <reference path="~/test/lib/qunit/qunit-1.12.0.js"/>
2   /// <reference path="~/src/Person.js"/>
3   /// <reference path="~/src/MyException.js"/>
4
5   var person = new Person('Jim', 22);
6   //module() - Group related tests under a single label.
7   module('ok');
8
9   //test() - Add a test to run.
10  //ok() - A boolean assertion, equivalent to CommonJS's asser
11  test("ok", function() {
12      ok(true, 'true is ok');
13      ok(person != null, 'Person is not null');
14  });
```

Run QUnit in R# 7

```
1   /// <reference path="~/test/lib/qunit/qunit-1.12.0.js"/>
2   /// <reference path="~/src/Person.js"/>
3   /// <reference path="~/src/MyException.js"/>
4
5   var person = new Person('Jim', 22);
6   //module() - Group related tests under a single label.
7   module('ok');
8
9   //test() - Add a test to run.
10  //ok() - A boolean assertion, equivalent to CommonJS's asser
11  test("ok", function() {
12      ok(true, 'true is ok');
13      ok(person != null, 'Person is not null');
14  });
```

Run QUnit in R# 7

```
1   /// <reference path="~/test/lib/qunit/qunit-1.12.0.js"/>
2   /// <reference path="~/src/Person.js"/>
3   /// <reference path="~/src/MyException.js"/>
4
5   var person = new Person('Jim', 22);
6   //module() - Group related tests under a single label.
7   module('ok');
8
9   //test() - Add a test to run.
10  //ok() - A boolean assertion, equivalent to CommonJS's asser
11  test("ok", function() {
12      ok(true, 'true is ok');
13      ok(person != null, 'Person is not null');
14  });
```

ok
▷ Run
🐞 Debug
✚ Append to Session ▸

√ ok #3 ✕

Group by: Projects and Namespaces ▼

🔗 9   √ 9   ⛔ 0   ∅ 0   ❓ 0

□ √ 🖥 <qunit> *(9 tests)*	Success
□ √ deepEqual and notDeepEqual *(2 tests)*	Success
√ deepEqual	Success: (5/0 of 5)
√ notDeepEqual	Success: (3/0 of 3)
□ √ equal and notEqual *(4 tests)*	Success
√ equal	Success: (6/0 of 6)
√ notEqual	Success: (4/0 of 4)
√ notStrictEqual	Success: (4/0 of 4)
√ scrictEqual	Success: (2/0 of 2)
□ √ expect *(1 test)*	Success
√ test withc expect	Success: (2/0 of 2)
□ √ ok *(1 test)*	Success
√ ok	Success
□ √ throws exception *(1 test)*	Success
√ throw "error"	Success: (3/0 of 3)

Task 4

- A browser JavaScript testing toolkit
- A Node.js testing toolkit
- Flexible
- Written by you
- A set of reusable libraries
- The future

http://docs.busterjs.org/

Installation

```
npm install buster
```

npm i **buster** --save-dev

Create config file

```
var config = module.exports;

config["My tests"] = {
    environment: "node", // or "browser"
    rootPath: "../",
    sources: [
    ],
    tests: [
        "test/*Test.js"
    ]
};
```

Write tests

```javascript
var buster = require('buster');
var assert = buster.assertions.assert;

buster.testCase('test', {
    'test 1': function() {
        assert.equals(2 + 3, 5);
    }
});
```

Write tests

```javascript
var buster = require('buster');
var assert = buster.assertions.assert;

buster.testCase('test', {
    'test 1': function() {
        assert.equals(2 + 3, 5);
    }
});
```

```javascript
'same': function() {
    var obj = { id: 42, name: "Chris" };
    assert.same(obj, obj);
},
'equals': function() {
    assert.equals(true, true, 'true equals true');
    assert.equals(1, 1, '1 equals 1');
    assert.equals('some', 'some', '"some" equals "some"');

    assert.equals([], [], '[] equals []');
    assert.equals([1, 2, 3], [1, 2, 3], '[1, 2, 3] equals [1, 2, 3]');
    assert.equals([[1], [2], [3]], [[1], [2], [3]], '[[1], [2], [3]] equals [[

    assert.equals({}, {}, '{} equals {}');
    assert.equals({foo: 'bar'}, {foo: 'bar'}, '{foo: \'bar\'} equals {foo: \'b

    assert.equals(null, null, 'null equals null');
    assert.equals(undefined, undefined, 'undefined equals undefined');
    assert.equals(undefined, undefined, 'undefined equals undefined');

    assert.equals(NaN, NaN, 'NaN equals NaN');
},
```

```javascript
'same': function() {
    var obj = { id: 42, name: "Chris" };
    assert.same(obj, obj);
},
'equals': function() {
    assert.equals(true, true, 'true equals true');
    assert.equals(1, 1, '1 equals 1');
    assert.equals('some', 'some', '"some" equals "some"');

    assert.equals([], [], '[] equals []');
    assert.equals([1, 2, 3], [1, 2, 3], '[1, 2, 3] equals [1, 2, 3]');
    assert.equals([[1], [2], [3]], [[1], [2], [3]], '[[1], [2], [3]] equals [[

    assert.equals({}, {}, '{} equals {}');
    assert.equals({foo: 'bar'}, {foo: 'bar'}, '{foo: \'bar\'} equals {foo: \'b

    assert.equals(null, null, 'null equals null');
    assert.equals(undefined, undefined, 'undefined equals undefined');
    assert.equals(undefined, undefined, 'undefined equals undefined');

    assert.equals(NaN, NaN, 'NaN equals NaN');
},
```

```javascript
'same': function() {
    var obj = { id: 42, name: "Chris" };
    assert.same(obj, obj);
},
'equals': function() {
    assert.equals(true, true, 'true equals true');
    assert.equals(1, 1, '1 equals 1');
    assert.equals('some', 'some', '"some" equals "some"');

    assert.equals([], [], '[] equals []');
    assert.equals([1, 2, 3], [1, 2, 3], '[1, 2, 3] equals [1, 2, 3]');
    assert.equals([[1], [2], [3]], [[1], [2], [3]], '[[1], [2], [3]] equals [[

    assert.equals({}, {}, '{} equals {}');
    assert.equals({foo: 'bar'}, {foo: 'bar'}, '{foo: \'bar\'} equals {foo: \'b

    assert.equals(null, null, 'null equals null');
    assert.equals(undefined, undefined, 'undefined equals undefined');


    assert.equals(NaN, NaN, 'NaN equals NaN');
},
```

```javascript
'matcher': function() {
    assert.match("Give me something", "Give");
    assert.match({ toString: function () { return "foo"; } }, "foo");

    assert.match(true, true);
    assert.match(false, false);

    assert.match("Give me something", /^([a-z]\s*)+$/i);
    assert.match({ toString: function () { return "yeah!"; } }, /yeah/);

    assert.match(5, 5);

    assert.match("123", function (exp) { return exp == '123'; });
    assert.match(
        {toString: function () { return '42'; }},
        function () { return true; });
```

```
    assert.match(
        '123',
        {test: function (arg) { return arg == 123;} });
    assert.match({
            name: 'Chris',
            profession: 'Programmer'
        }, {
            name: 'Chris'
        });
    },
```

```javascript
'isObject': function() {
    assert.isObject({});
    assert.isObject([1, 2, 3]);
},
'isFunction': function() {
    assert.isFunction(function () {});
},
'isTrue': function() {
    assert.isTrue(true);
    assert.isTrue(1 == true);
    assert.isTrue(null == undefined);
},
'isFalse': function() {
    assert.isFalse(false);
    assert.isFalse(1 === true);
    assert.isFalse(null === undefined);
},
```

```javascript
'isString': function() {
    assert.isString('');
},
'isBoolean': function() {
    assert.isBoolean(true);
    assert.isBoolean(false);
},
'isNumber': function() {
    assert.isNumber(543);
},
'isNaN': function() {
    assert.isNaN(NaN);
},
'isArray': function() {
    assert.isArray([]);
    assert.isArray(Array());
},
'isArrayLike': function() {
    assert.isArrayLike([1, 2, 3]);
    assert.isArrayLike(arguments);
    assert.isArrayLike({ length: 0, splice: function() {} });
},
```

```javascript
'exception': function() {
    assert.exception(function () {
        throw new Error('Ooops!');
    });

    assert.exception(function () {
        throw new TypeError('Ooops!');
    }, 'TypeError');
},
'near': function() {
    assert.near(10.3, 10, 0.5);
    assert.near(10.5, 10, 0.5);
},
'hasPrototype': function() {
    assert.hasPrototype(function() {}, Function.prototype);
    assert.hasPrototype(function() {}, Object.prototype);
},
'contains': function() {
    assert.contains([1, 2, 3], 2);
    assert.contains('abc', 'a');
}
```

Task 5

Karma

Spectacular Test Runner for JavaScript

On the AngularJS team, we rely on testing and we always seek better tools to make our life easier.

Installation

`npm install ` **`karma`**

`npm i ` **`karma`** ` --save-dev`

Create config file

```javascript
module.exports = function(config) {
  config.set({
    basePath :   './',
    frameworks: ['jasmine'],
    files : [
      //'node_modules/should/should.js',
      //'test/*_mocha.js'
      'test/*_jasmine.js'
      //'test/*_qunit.js'
    ],
    exclude: [   ],
    reporters: ['progress'],
    port: 9876,
    colors: true,
    logLevel: config.LOG_INFO,
    autoWatch: true,
    browsers: ['PhantomJS'],
    captureTimeout: 60000,
    singleRun: false
  });
};
```

Karma

Task 6

Testem

A test runner that makes JavaScript unit testing fun.

Unit testing in JavaScript can be tedious and painful, but Testem makes it so easy that you will actually want to write tests.

Features

- Test-framework agnostic.
  - Jasmine
  - Qunit
  - Mocha
  - Buster
- Run tests in all major browsers, Node, PhantomJS
- Two distinct use-cases
  - Test-Driven-Development
  - Continuous Integration
- ...

Installation

```
npm install testem
```

```
npm i testem --save-dev
```

Create config file

```json
{
    "framework": "jasmine",
    "src_files": [
        "test/*.js"
    ]
}
```

testem --help

```
Usage: testem.js [options]

Commands:

  launchers              Print the list of available launchers (browsers & process launchers)
  ci [options]           Continuous integration mode
  server                 Run just the server

Options:

  -h, --help             output usage information
  -V, --version          output the version number
  -f, --file [file]      config file - defaults to testem.json or testem.yml
  -p, --port [num]       server port - defaults to 7357
  --host [hostname]      host name - defaults to localhost
  -l, --launch [list]    list of launchers to launch(comma separated)
  -s, --skip [list]      list of launchers to skip(comma separated)
  -d, --debug            output debug to debug log - testem.log
  -t, --test_page [page] the html page to drive the tests
  -g, --growl            turn on growl notifications

Keyboard Controls (in dev mode):

  ENTER                  run the tests
  q                      quit
  LEFT ARROW             move to the next browser tab on the left
  RIGHT ARROW            move to the next browser tab on the right
  TAB                    switch between top and bottom panel (split mode only)
  UP ARROW               scroll up in the target text panel
  DOWN ARROW             scroll down in the target text panel
  SPACE                  page down in the target text panel
  b                      page up in the target text panel
  d                      half a page down in the target text panel
  u                      half a page up in the target text panel
```

Testem (0)

```
TEST'EM 'SCRIPTS!
Open the URL below in a browser to connect.
http://localhost:7357/
-------------+
  Firefox 24.0|
    0/0 v     |
              +-----------------------------------------------------------
No tests were run :(
```

```
TEST'EM 'SCRIPTS!
Open the URL below in a browser to connect.
http://localhost:7357/
-------------+
  Firefox 24.0|
    3/3 v     |
              +-----------------------------------------------------------
v 3 tests complete.
```

Testem (1)


```
TEST'EM 'SCRIPTS!
Open the URL below in a browser to connect.
http://localhost:7357/
-------------+
  Firefox 24.0|
    2/3 x     |
             +------------------------------------------------------------------
kata calculator empty string equals 0.
    x Expected NaN to be 0.
        jasmine.ExpectationResult@http://localhost:7357/testem/jasmine.js:114
        jasmine.Matchers.matcherFn_/<@http://localhost:7357/testem/jasmine.js:1240
        @http://localhost:7357/test%5CCalculatorTest_jasmine.js:9
        jasmine.Block.prototype.execute@http://localhost:7357/testem/jasmine.js:1064
        jasmine.Queue.prototype.next_@http://localhost:7357/testem/jasmine.js:2096
        jasmine.Queue.prototype.start@http://localhost:7357/testem/jasmine.js:2049
        jasmine.Spec.prototype.execute@http://localhost:7357/testem/jasmine.js:2376
        jasmine.Queue.prototype.next_@http://localhost:7357/testem/jasmine.js:2096
        jasmine.Queue.prototype.start@http://localhost:7357/testem/jasmine.js:2049
        jasmine.Suite.prototype.execute@http://localhost:7357/testem/jasmine.js:2521
        jasmine.Queue.prototype.next_@http://localhost:7357/testem/jasmine.js:2096
        jasmine.Queue.prototype.start@http://localhost:7357/testem/jasmine.js:2049
        jasmine.Runner.prototype.execute@http://localhost:7357/testem/jasmine.js:214
3
        jasmine.Env.prototype.execute@http://localhost:7357/testem/jasmine.js:802
        window.onload@http://localhost:7357/6515:13
```

Task 7

In summary

- Today JS is no longer the „*add some animation to my website*" language.

- It's now the language of the web.

- JS code can and should be tested.

- There are many tools that make testing easy and enjoyable in JavaScript.

Resource

- **Test-Driven JavaScript Development** by *Christian Johansen*.
- **JavaScript Unit Testing** by *Hazem Saleh*.
- **Testable JavaScript** by *Mark Ethan Trostler*.

- http://pivotal.github.io/jasmine/
- http://visionmedia.github.io/mocha/
- http://qunitjs.com
- http://busterjs.org
- http://karma-runner.github.io
- https://github.com/airportyh/testem

Examples

All examples of this presentation
(and even more) are available at

https://**github**.com/**slon1024/workshop_test_js**