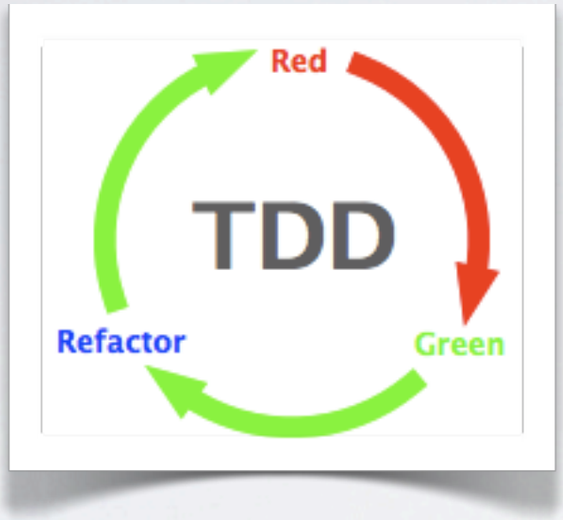
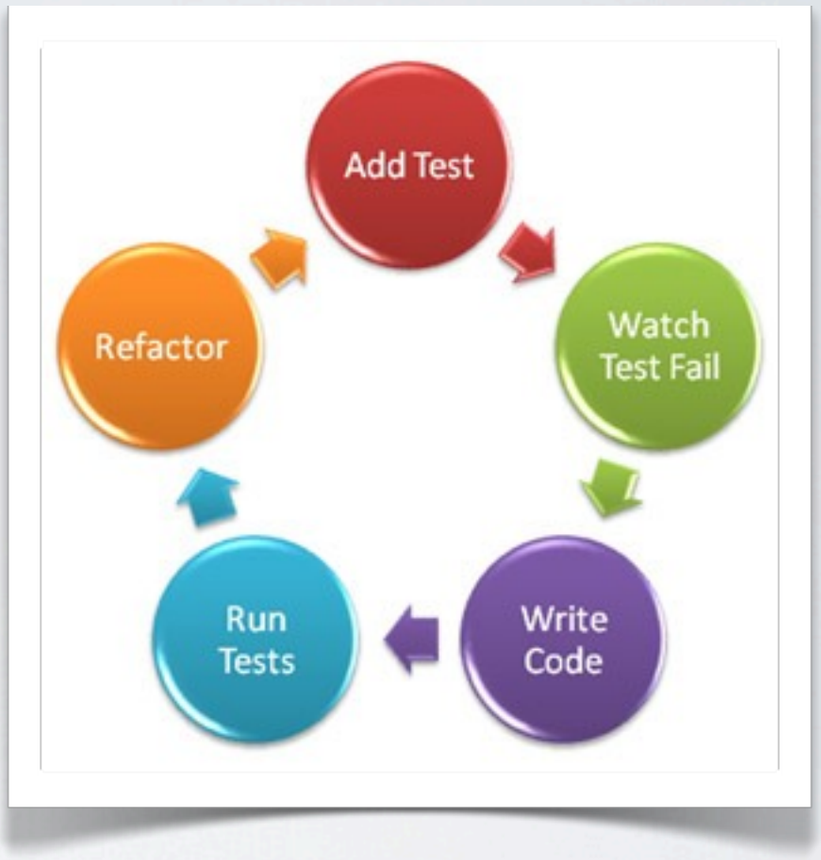
BDD for your JavaScript

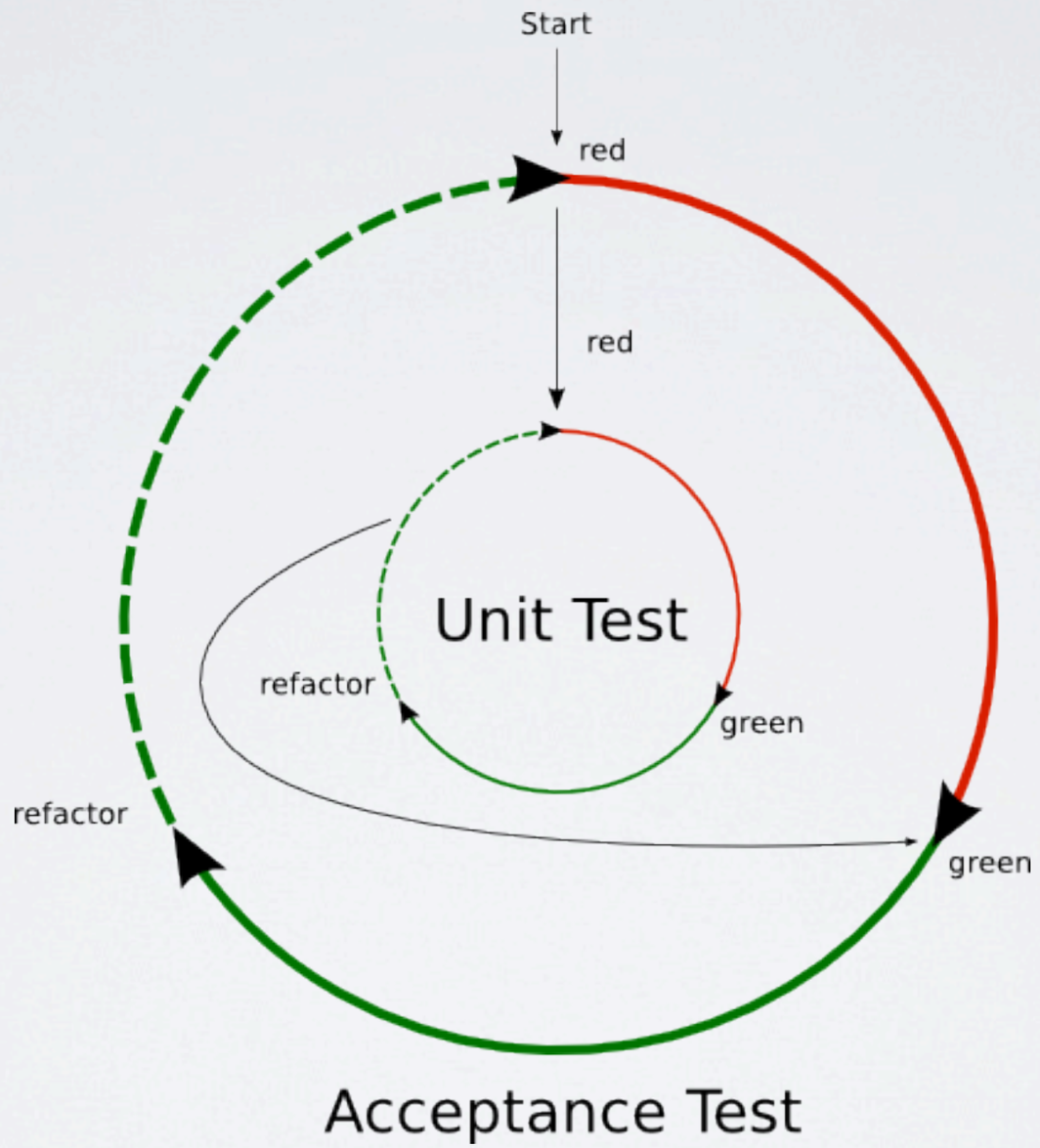
Jasmine is a behavior-driven development framework for testing your JavaScript code. It does not depend on any other JavaScript frameworks. It does not require a DOM. And it has a clean, obvious syntax so that you can easily write tests.

What is TDD?


Red
TDD
Refactor
Green


Add Test
Watch Test Fail
Write Code
Run Tests
Refactor

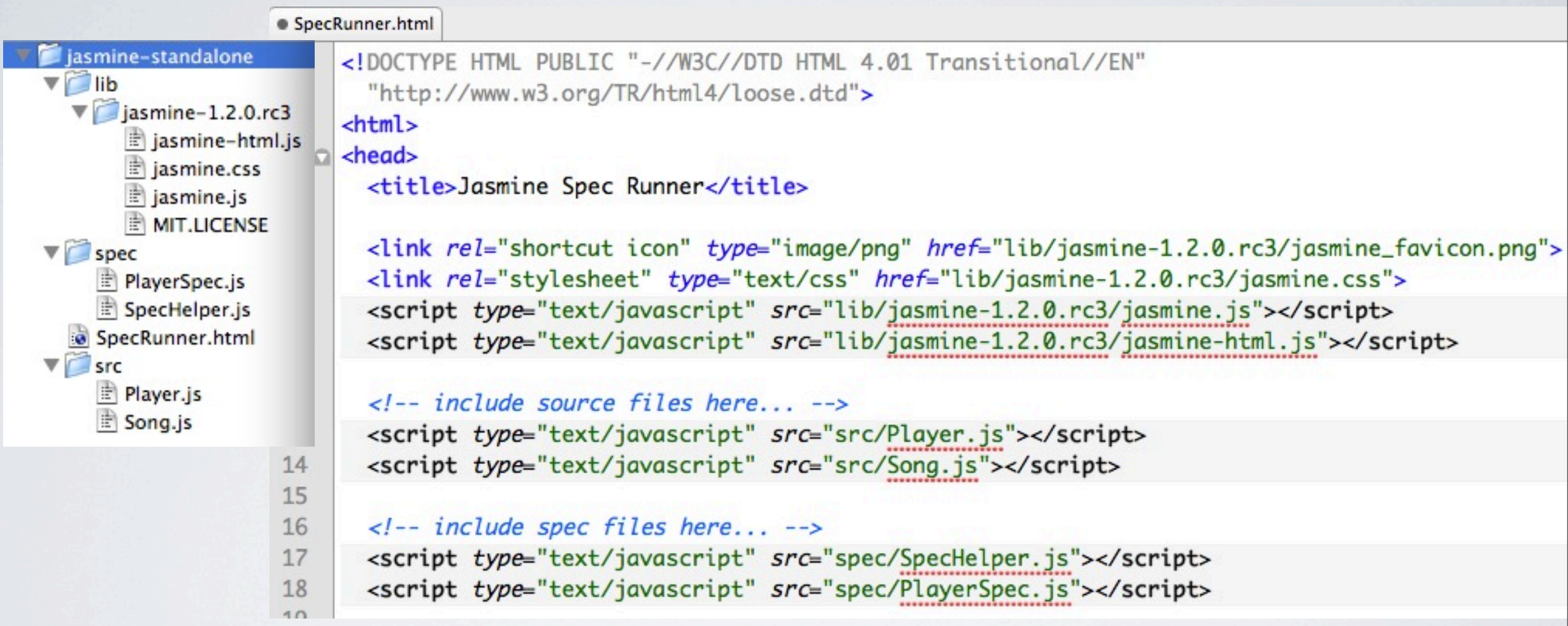
What is BDD?


Start
red
red
Unit Test
refactor
green
refactor
green
Acceptance Test

I can use Jasmine with:

Standalone for simple, browser page or console	*http://pivotal.github.com/jasmine/download.html*
Node.js	*npm install jasmine-node -g* *https://github.com/mhevery/jasmine-node*
Ruby WEB project or into Rails project	*gem install jasmine* *https://github.com/pivotal/jasmine-gem*
Java (with Maven)	*http://searls.github.com/jasmine-maven-plugin/*
Django	*https://github.com/Fandekasp/django-jasmine*
.NET	*http://lostechies.com/josharnold/2012/02/25/running-jasmine-specs-in-dotnet-with-serenity/*

Standalone


SpecRunner.html
jasmine-standalone
lib
jasmine-1.2.0.rc3
jasmine-html.js
jasmine.css
jasmine.js
MIT.LICENSE
spec
PlayerSpec.js
SpecHelper.js
SpecRunner.html
src
Player.js
Song.js
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Jasmine Spec Runner</title>
<link rel="shortcut icon" type="image/png" href="lib/jasmine-1.2.0.rc3/jasmine_favicon.png">
<link rel="stylesheet" type="text/css" href="lib/jasmine-1.2.0.rc3/jasmine.css">
<script type="text/javascript" src="lib/jasmine-1.2.0.rc3/jasmine.js"></script>
<script type="text/javascript" src="lib/jasmine-1.2.0.rc3/jasmine-html.js"></script>
<!-- include source files here... -->
<script type="text/javascript" src="src/Player.js"></script>
<script type="text/javascript" src="src/Song.js"></script>
<!-- include spec files here... -->
<script type="text/javascript" src="spec/SpecHelper.js"></script>
<script type="text/javascript" src="spec/PlayerSpec.js"></script>
14
15
16
17
18

Green tests

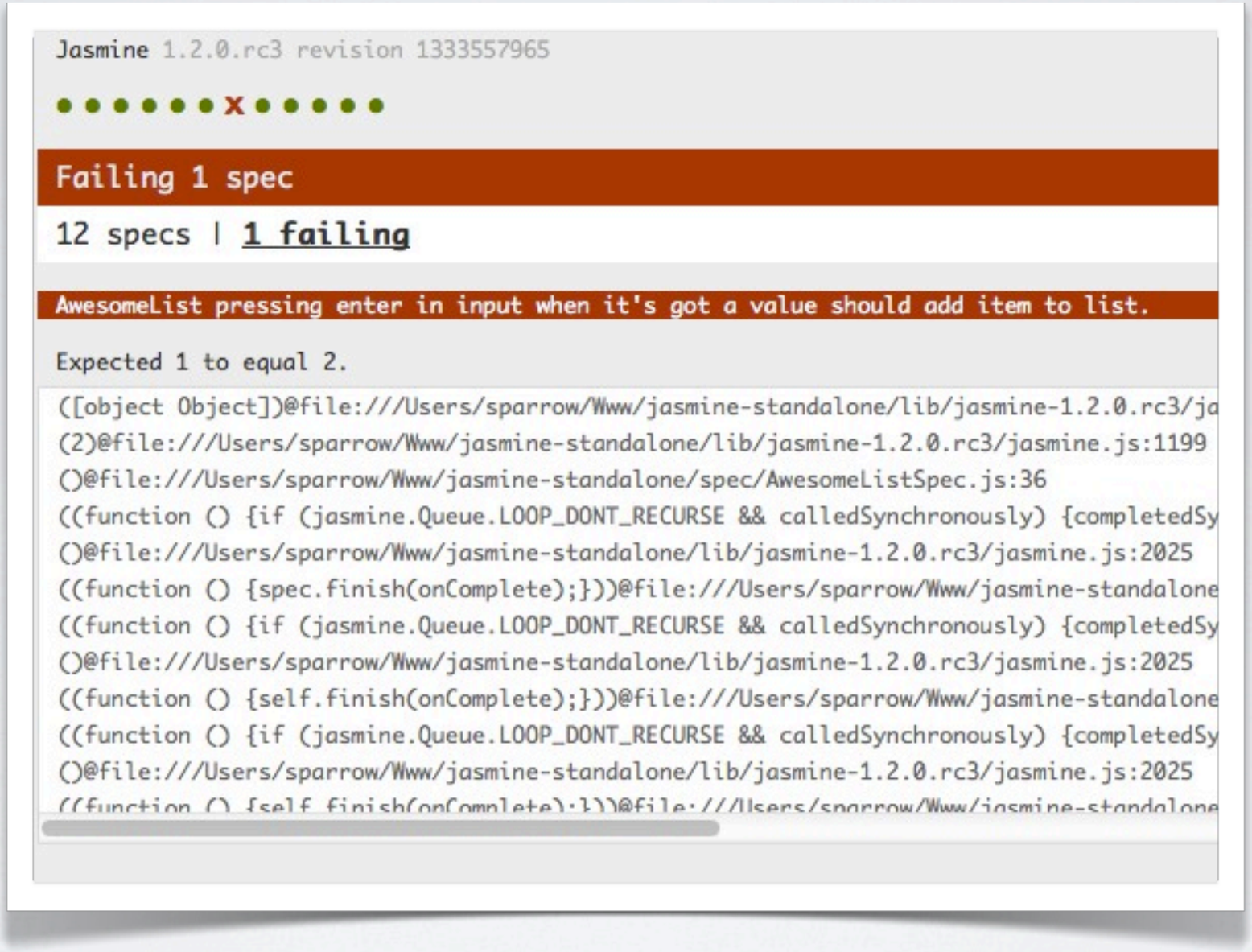
Jasmine 1.2.0.rc3 revision 1333557965

●●●●●●●●●●●●

Passing 12 specs

```
Player
  should be able to play a Song

  when song has been paused
    should indicate that the song is currently paused
    should be possible to resume
  tells the current song if the user has made it a favorite

  #resume
    should throw an exception if song is already playing

AwesomeList
  pressing enter in input when there's no value
    should not add an item

  pressing enter in input when it's got a value
    should add item to list
    item should contain entered text

  clicking remove link
    should remove item

Spy
  should spy on a static method of Klass
  should spy on an instance method of a Klass
  should spy on Klass#methodWithCallback
```

Red tests


Jasmine 1.2.0.rc3 revision 1333557965
Failing 1 spec
12 specs | 1 failing
AwesomeList pressing enter in input when it's got a value should add item to list.
Expected 1 to equal 2.
([object Object])@file:///Users/sparrow/Www/jasmine-standalone/lib/jasmine-1.2.0.rc3/ja
(2)@file:///Users/sparrow/Www/jasmine-standalone/lib/jasmine-1.2.0.rc3/jasmine.js:1199
()@file:///Users/sparrow/Www/jasmine-standalone/spec/AwesomeListSpec.js:36
((function () {if (jasmine.Queue.LOOP_DONT_RECURSE && calledSynchronously) {completedSy
()@file:///Users/sparrow/Www/jasmine-standalone/lib/jasmine-1.2.0.rc3/jasmine.js:2025
((function () {spec.finish(onComplete);}))@file:///Users/sparrow/Www/jasmine-standalone
((function () {if (jasmine.Queue.LOOP_DONT_RECURSE && calledSynchronously) {completedSy
()@file:///Users/sparrow/Www/jasmine-standalone/lib/jasmine-1.2.0.rc3/jasmine.js:2025
((function () {self.finish(onComplete);}))@file:///Users/sparrow/Www/jasmine-standalone
((function () {if (jasmine.Queue.LOOP_DONT_RECURSE && calledSynchronously) {completedSy
()@file:///Users/sparrow/Www/jasmine-standalone/lib/jasmine-1.2.0.rc3/jasmine.js:2025

Specs

```
// it() is JavaScript function with a description string and the function
it('describes the current test', function () {
  // some JS code
});
```

Expectations

```
it('should increment a variable', function () {
  var plusOne = function(v){ return v + 1 }; // set up the world
  var result = plusOne(0);                   // call your application code
  expect(result).toEqual(1);                 // verifying the expected behavior
});
```

Group tests with ‘describe’

```
describe('Calculator', function () {
  it('can add a number', function () {
    //...
  });

  it('can multiply some numbers', function () {
    //...
  });

  describe('can divide some numbers', function () {
    it('should not divide on zero', function () {
      //...
    });
  });
});
```

beforeEach

```
describe('some suite', function () {
  var suiteWideFoo = 0;

  beforeEach(function () {
    suiteWideFoo = 1;
  });

  it('should equal bar', function () {
    expect(suiteWideFoo).toEqual(1);
  });
});
```

beforeEach

```
var runnerWideFoo = [];

beforeEach(function () {
  runnerWideFoo.push('runner');
});

describe('some suite', function () {
  beforeEach(function () {
    runnerWideFoo.push('some');
  });

  it('should equal [runner, some]', function () {
    expect(runnerWideFoo).toEqual(['runner', 'some']);
  });

  describe('another suite', function () {
    beforeEach(function () {
      runnerWideFoo.push('another');
    });

    it('should equal [runner, some, runner, some, another]', function () {
      expect(runnerWideFoo).toEqual([ 'runner', 'some', 'runner', 'some', 'another' ]);
    });
  });
});
```

afterEach

```
describe('some suite', function () {
  var suiteWideFoo = 1;
  afterEach(function () {
    suiteWideFoo = 0;
  });

  it('should equal 1', function () {
    expect(suiteWideFoo).toEqual(1);
  });

  it('should equal 0 after', function () {
    expect(suiteWideFoo).toEqual(0);
  });
});
```

Matchers

```
expect(x).toEqual(y);                // compares objects or primitives 'x' and 'y' and passes if they are equivalent
expect(x).toBe(y);                   // compares objects or primitives 'x' and 'y' and passes if they are the same object
expect(x).toMatch(pattern);          // compares 'x' to string or regular expression 'pattern' and passes if they match
expect(x).toBeDefined();             // passes if 'x' is not undefined
expect(x).toBeUndefined();           // passes if 'x' is undefined
expect(x).toBeNull();                // passes if 'x' is null
expect(x).toBeTruthy();              // passes if 'x' evaluates to true
expect(x).toBeFalsy();               // passes if 'x' evaluates to false
expect(x).toContain(y);              // passes if array or string 'x' contains 'y'
expect(x).toBeLessThan(y);           // passes if 'x' is less than 'y'
expect(x).toBeGreaterThan(y);        // passes if 'x' is greater than 'y'
expect(function(){fn();}).toThrow(e); // passes if function 'fn' throws exception 'e' when executed
```

Writing Own Matchers

```
describe("Create own Matcher", function() {
  beforeEach(function() {
    this.addMatchers({
      toBeBetween: function(from, to) {
        return this.actual > from && this.actual < to;
      }
    });
  });

  it('5 should be between 4 and 6', function () {
    expect(5).toBeBetween(4, 6);
  });
});
```

Spies

Mocks and Stubs

goto src

Spy Matchers

```
expect(x).toHaveBeenCalled()                    // passes if 'x' is a spy and was called
expect(x).toHaveBeenCalledWith(arguments)       // passes if x is a spy and was called with the specified arguments
expect(x).not.toHaveBeenCalled()                // passes if x is a spy and was not called
expect(x).not.toHaveBeenCalledWith(arguments) // passes if x is a spy and was not called with the specified arguments

spyOn(x, 'method').andCallThrough()       // spies on AND calls the original function spied on
spyOn(x, 'method').andReturn(arguments)   // returns passed arguments when spy is called
spyOn(x, 'method').andThrow(exception)    // throws passed exception when spy is called
spyOn(x, 'method').andCallFake(function) // calls passed function when spy is called
```

Sinon.JS

http://sinonjs.org/

How to test jQuery plugins

+

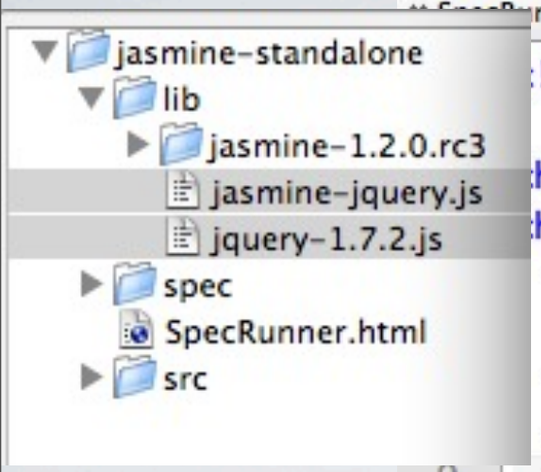
= jasmine-jquery

https://github.com/velesin/jasmine-jquery

Jasmine-jquery provides two extensions for Jasmine:

- a set of custom matchers for jQuery framework
- an API for handling HTML fixtures in your specs

Add jasmine-jquery to test env



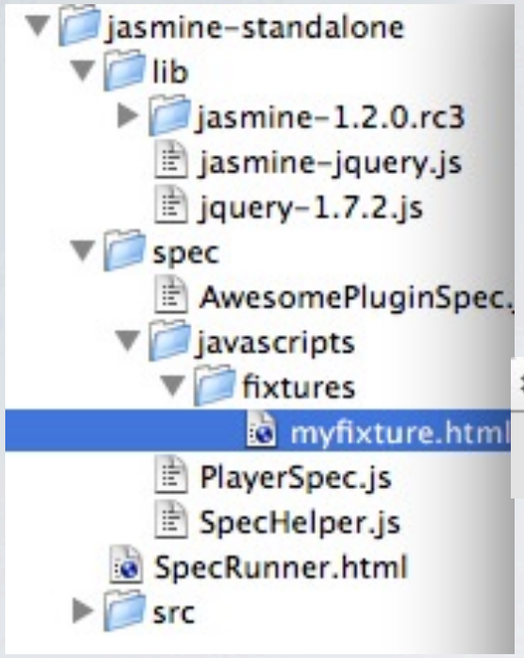
SpecRunner.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <title>Jasmine Spec Runner</title>

  <link rel="shortcut icon" type="image/png" href="lib/jasmine-1.2.0.rc3/jasmine_favicon.png">
  <link rel="stylesheet" type="text/css" href="lib/jasmine-1.2.0.rc3/jasmine.css">
  <script type="text/javascript" src="lib/jasmine-1.2.0.rc3/jasmine.js"></script>
  <script type="text/javascript" src="lib/jasmine-1.2.0.rc3/jasmine-html.js"></script>

  <script type="text/javascript" src="lib/jquery-1.7.2.js"></script>
  <script type="text/javascript" src="lib/jasmine-jquery.js"></script>

  <!-- include source files here... -->
  <script type="text/javascript" src="src/Player.js"></script>
  <script type="text/javascript" src="src/Song.js"></script>

```

Some new Matchers

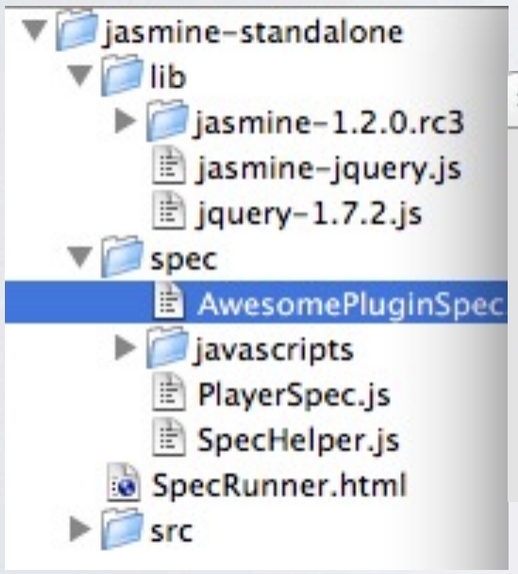
```
expect($('<div id="some-id"></div>'))                    .toBe('div#some-id')
expect($('<input type="checkbox" checked="checked"/>')) .toBeChecked()
expect($('<option selected="selected"></option>'))       .toBeSelected()
expect($('<div><span class="some-class"></span></div>')).toContain('span.some-class')
expect($('<div class="some-class"></div>'))              .toHaveClass("some-class")
expect($('<div><span></span></div>'))                    .toHaveHtml('<span></span>')
expect($('<div id="some-id"></div>'))                    .toHaveId("some-id")
expect($('<div>some text</div>'))                        .toHaveText('some text')
expect($('<input type="text" value="some text"/>'))      .toHaveValue('some text')
expect('<input type='submit' disabled='disabled'/>')     .toBeDisabled()
expect($('<input type='text' />').focus())               .toBeFocused()
```

Fixtures



myfixture.html

```
<div id="my-fixture">some complex content here</div>
```

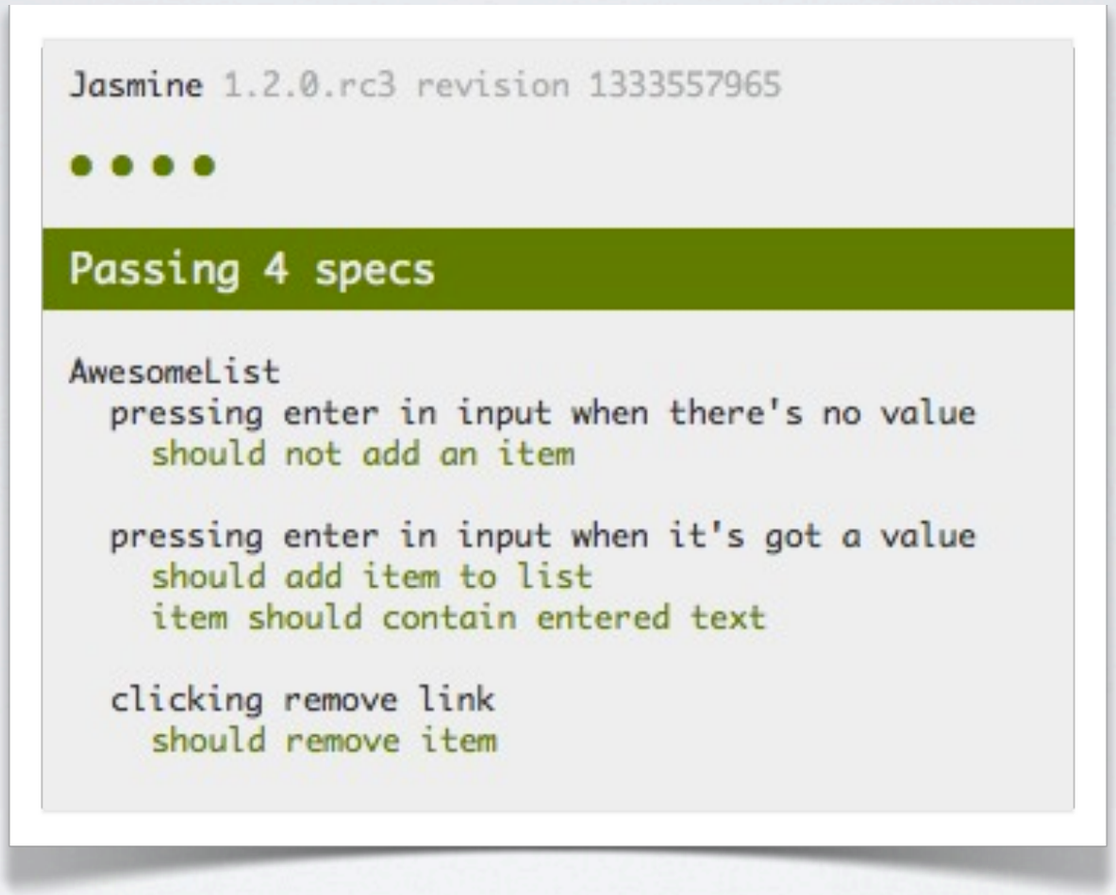


AwesomePluginSpec.js

```
describe("AwesomePlugin", function() {
  it('should have awesome behavior', function () {
    loadFixtures('myfixture.html');
    $('#my-fixture').myAwesomePlugin();
    expect($('#my-fixture')).toRuTuTuTu();
  });
});
```

Test jQuery plugin example

goto src



Questions