

# Testing JavaScript with Jasmine



By: Tim Tyrrell

# Tim Tyrrell



<http://willcode4beer.net>

[@timtyrrell](#)

# Agenda

- Briefly cover TDD
- Discuss what Jasmine is and isn't
- Show syntax with comparisons to RSpec
- Jasmine with:
  - Vanilla JavaScript
  - JavaScript through a rubygem
  - Jasmine with jQuery
  - Jasmine with Rails
  - Evergreen
- Where does CoffeeScript fit in?
- node.js, Maven, etc.
- Other helpful libraries

# Why unit test JavaScript?

# TDD?

"Red, Green, Refactor"

"There is only one way to go truly fast. Do the best job you can. Anything else is slower."

<http://twitter.com/#!/unclebobmartin/status/39438225869254656>

**I DON'T ALWAYS TEST MY  
CODE**



**BUT WHEN I DO I DO IT IN  
PRODUCTION**

Why Jasmine?



Jasmine is not an integration testing framework.

# RSpec vs. Jasmine: Structure

```
#RSpec
describe "Calculate" do
  describe "#add" do
    it "should return 2" do
      ...
    end
  end
end
```

```
//Jasmine
describe "Calculate", function(){
  describe "#Add", function(){
    it "should return 2", function(){
      ...
    };
  });
};
```

# RSpec vs. Jasmine: Before/After

```
#RSpec
before(:each) do
  @calc = Calculator.new
end
```

```
after(:each) do
  @calc.reset
end
```

```
//Jasmine
var calc;
beforeEach(function()
{   calc = new Calculator();
});
```

```
afterEach(function()
{   calc.reset();
});
```

# RSpec vs. Jasmine: Expectations

# RSpec

it "should add the numbers" do

calc = Calculator.new

calc.add(1,1).should == 2

calc.add(1,2).should\_not == 2 #Use one expectation per 'it'!

end

// Jasmine

it("should add the numbers", function() {

var calc = new Calculator();

expect(calc.Add(1,1)).toEqual(2);

expect(calc.Add(1,2)).not.toEqual(2); //Use one expectation per 'it'!

});

# Jasmine Matchers

`expect(x).toEqual(y)`

`expect(x).toBe(y);`

`expect(x).toMatch(pattern)`

`expect(x).toBeDefined()`

`expect(x).toBeNull();`

`expect(x).toBeTruthy();`

`expect(x).toBeFalsy();`

`expect(x).toContain(y);`

`expect(x).toBeLessThan(y);`

`expect(x).toBeGreaterThan(y);`

`expect(fn).toThrow(e);`

# RSpec vs. Jasmine: Stubbing

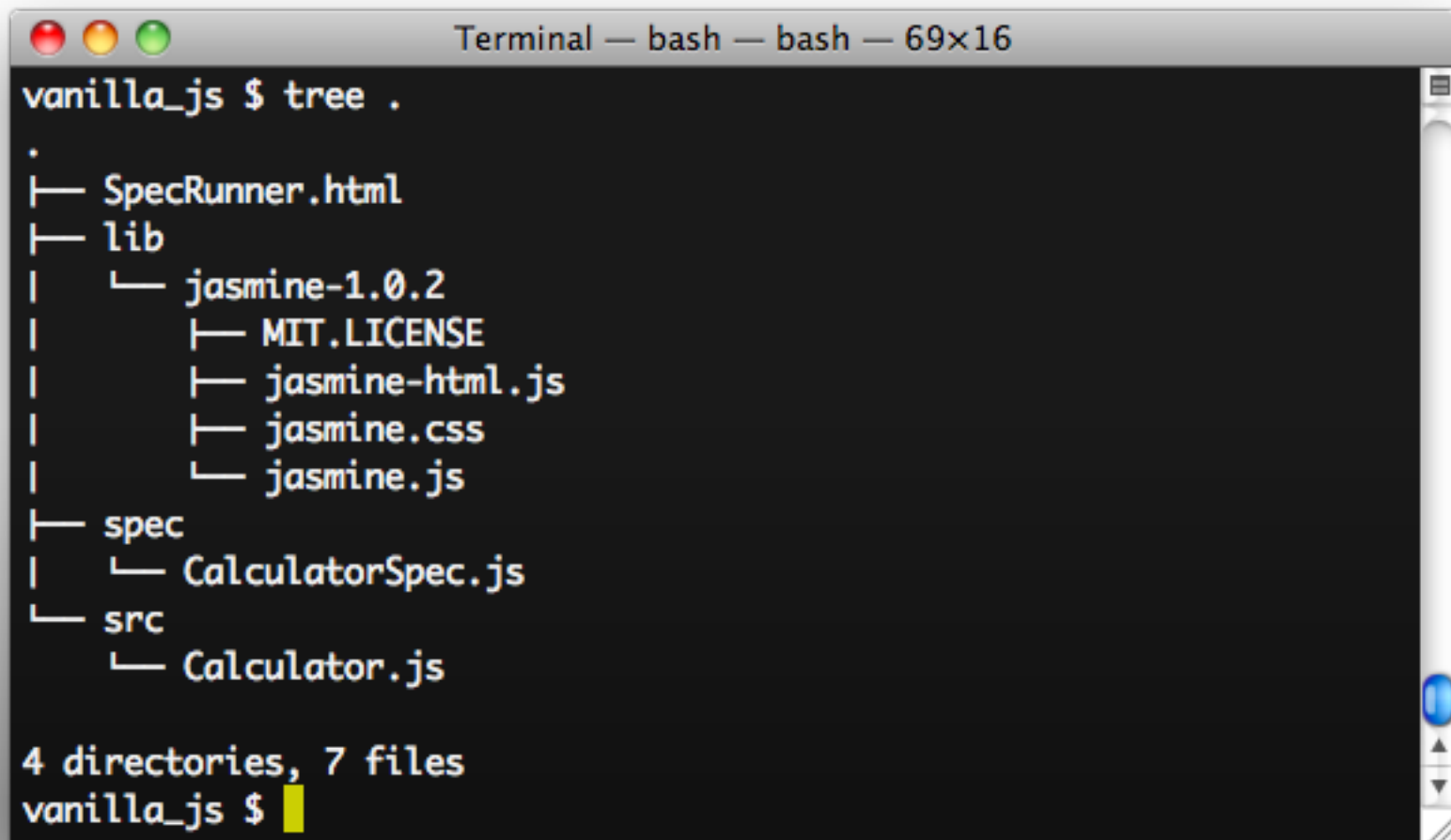
```
# RSpec
it "should add the numbers" do
  calc = Calculator.new
  calc.stub!(:add).and_return(3)
  calc.add(1,1).should == 3
end
```

```
// Jasmine
it("should add the numbers", function() {
  var calc = new Calculator();
  spyOn(calc, 'add').andReturn(3);
  expect(calc.Add(1,1)).toEqual(3);
});
```

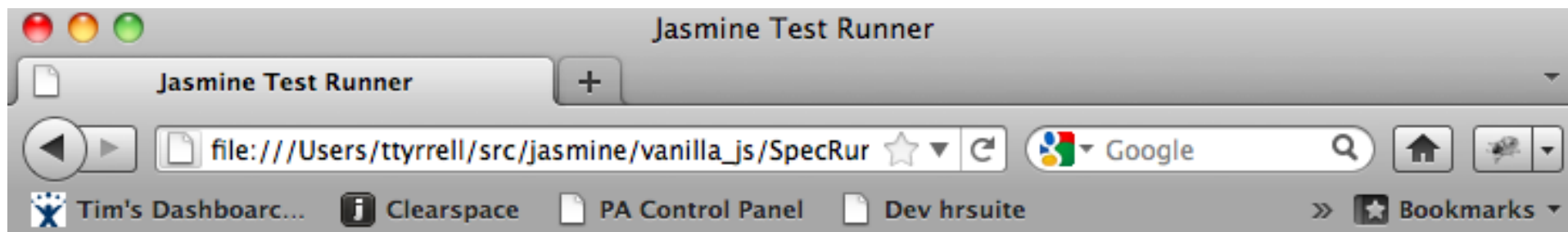
# Jasmine with "vanilla" JavaScript

- Download "Stand alone"

<http://pivotal.github.com/jasmine/download.html>



```
Terminal — bash — bash — 69x16
vanilla_js $ tree .
.
├── SpecRunner.html
├── lib
│   ├── jasmine-1.0.2
│   │   ├── MIT.LICENSE
│   │   ├── jasmine-html.js
│   │   ├── jasmine.css
│   │   └── jasmine.js
│   ├── spec
│   │   └── CalculatorSpec.js
│   └── src
│       └── Calculator.js
4 directories, 7 files
vanilla_js $
```



Jasmine 1.0.2 revision 1298837858

Show ☐ passed ☐ skipped

0 specs, 0 failures in 0s Finished at Sun Jun 19 2011 22:27:23 GMT-0500 (CDT)

run all



# Spec for Calculator:

```
describe("Calculator", function() {  
  var calc;
```

```
    beforeEach(function(){  
      calc = new Calculator();  
    });
```

```
    it("should add two numbers together", function() {  
      expect(calc.Add(1,1)).toEqual(2);  
    });  
  });
```

# Spec for Calculator:

```
describe("Calculator", function() {  
  var calc;
```

```
    beforeEach(function(){  
      calc = new Calculator();  
    });
```

```
    it("should add two numbers together", function() {  
      expect(calc.Add(1,1)).toEqual(2);  
    });  
  });
```

# Spec for Calculator:

```
describe("Calculator", function() {  
  var calc;
```

```
    beforeEach(function(){  
      calc = new Calculator();  
    });
```

```
    it("should add two numbers together", function() {  
      expect(calc.Add(1,1)).toEqual(2);  
    });  
  });
```

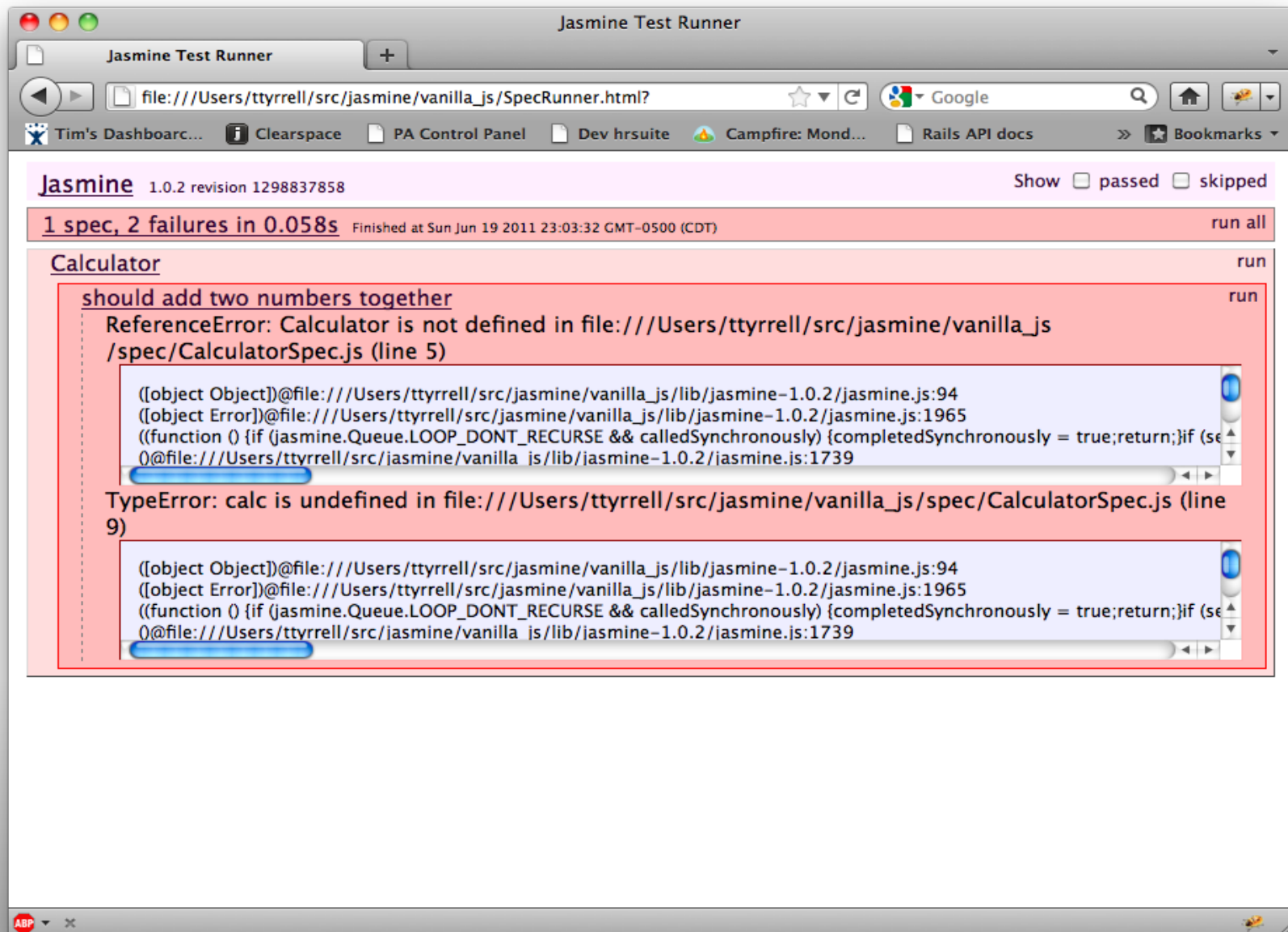
# Spec for Calculator:

```
describe("Calculator", function() {  
  var calc;
```

```
    beforeEach(function(){  
      calc = new Calculator();  
    });
```

```
    it("should add two numbers together", function() {  
      expect(calc.Add(1,1)).toEqual(2);  
    });  
  });
```

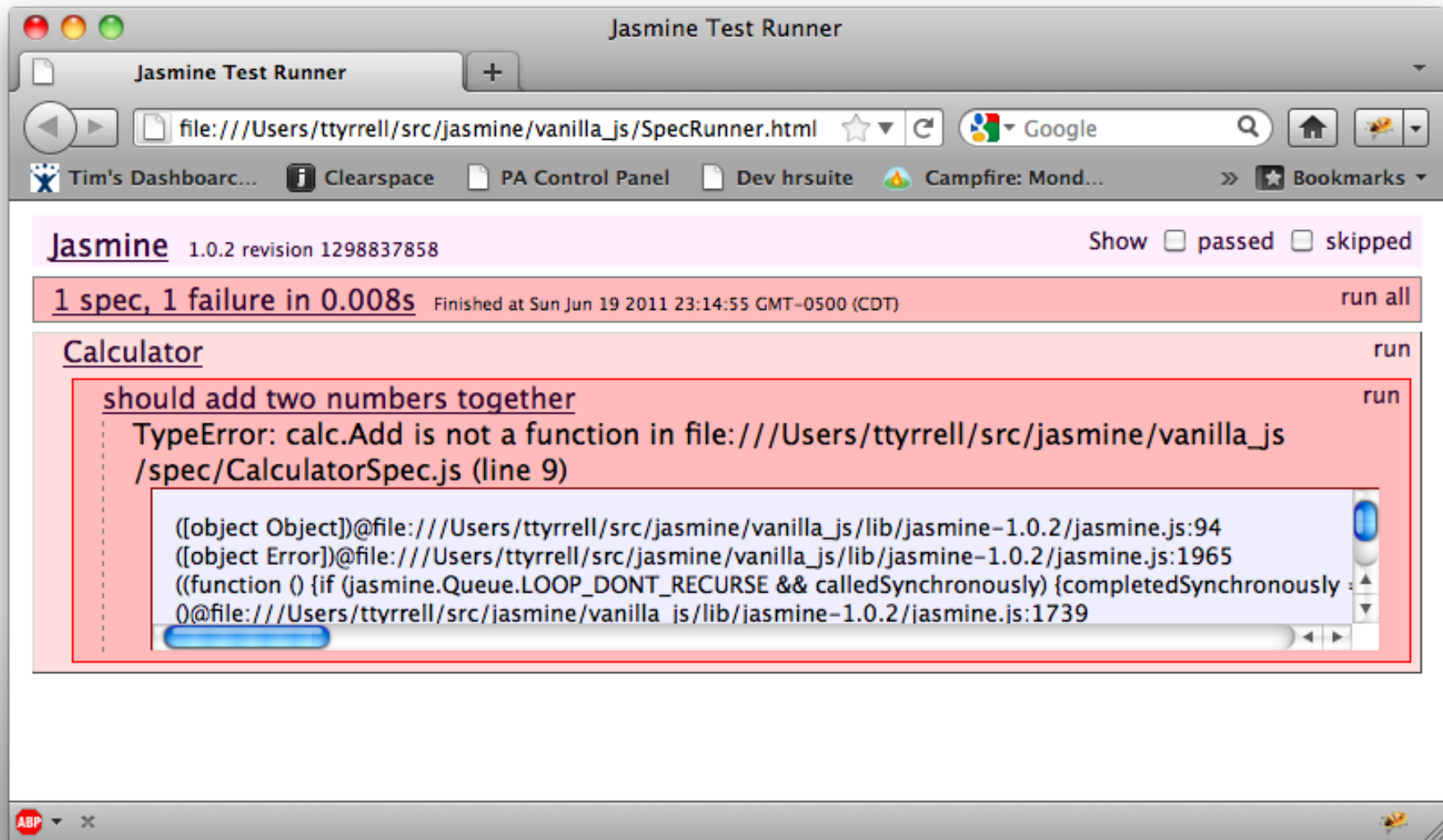
# Failing Test



# Stub out the "Calculator" function

```
function Calculator () {  
  
}
```

# One passing, one left to fix

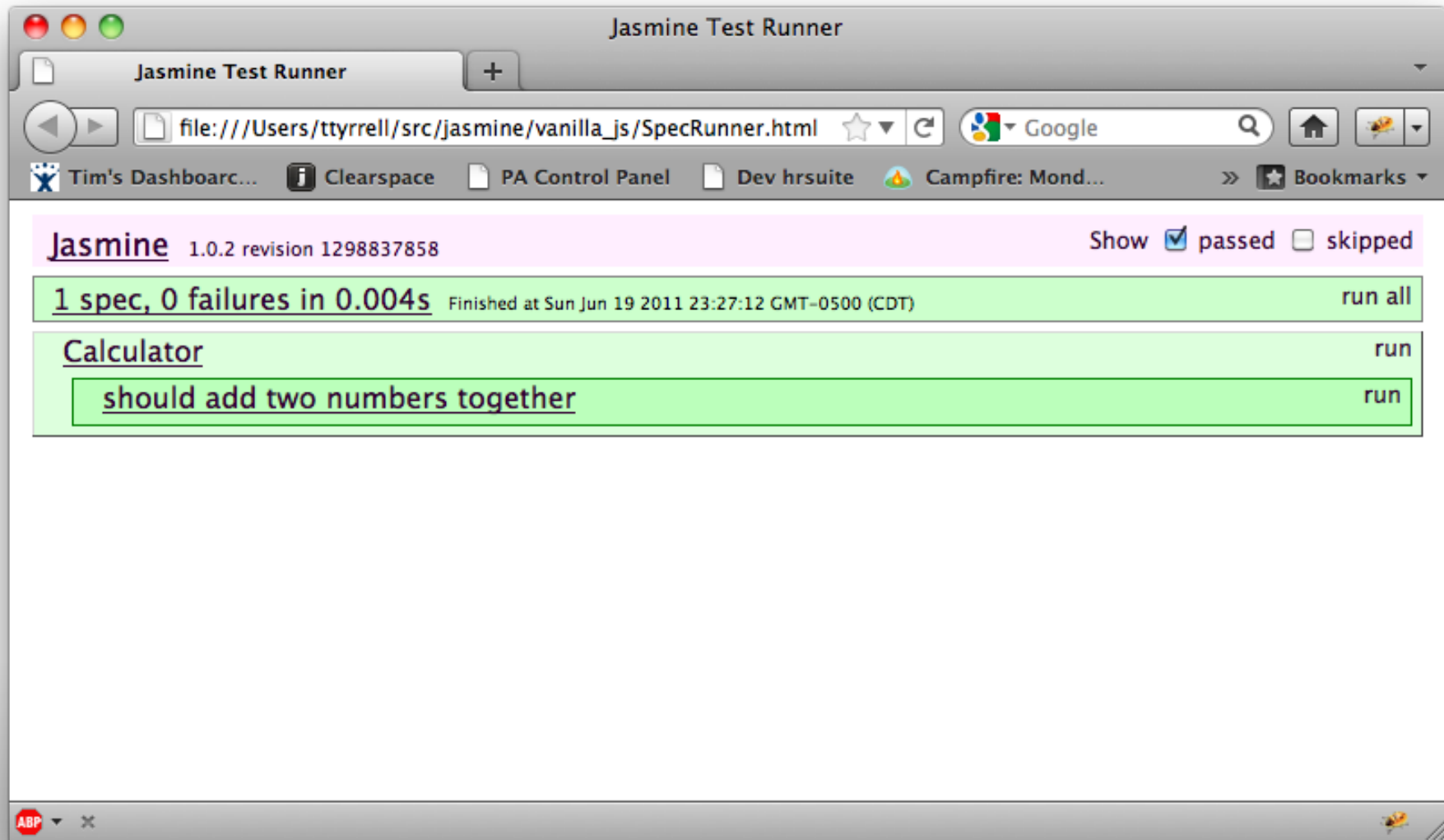


# Add the method

```
function Calculator () {  
  this.Add = function(num1, num2){  
    return num1 + num1;  
  };  
}
```



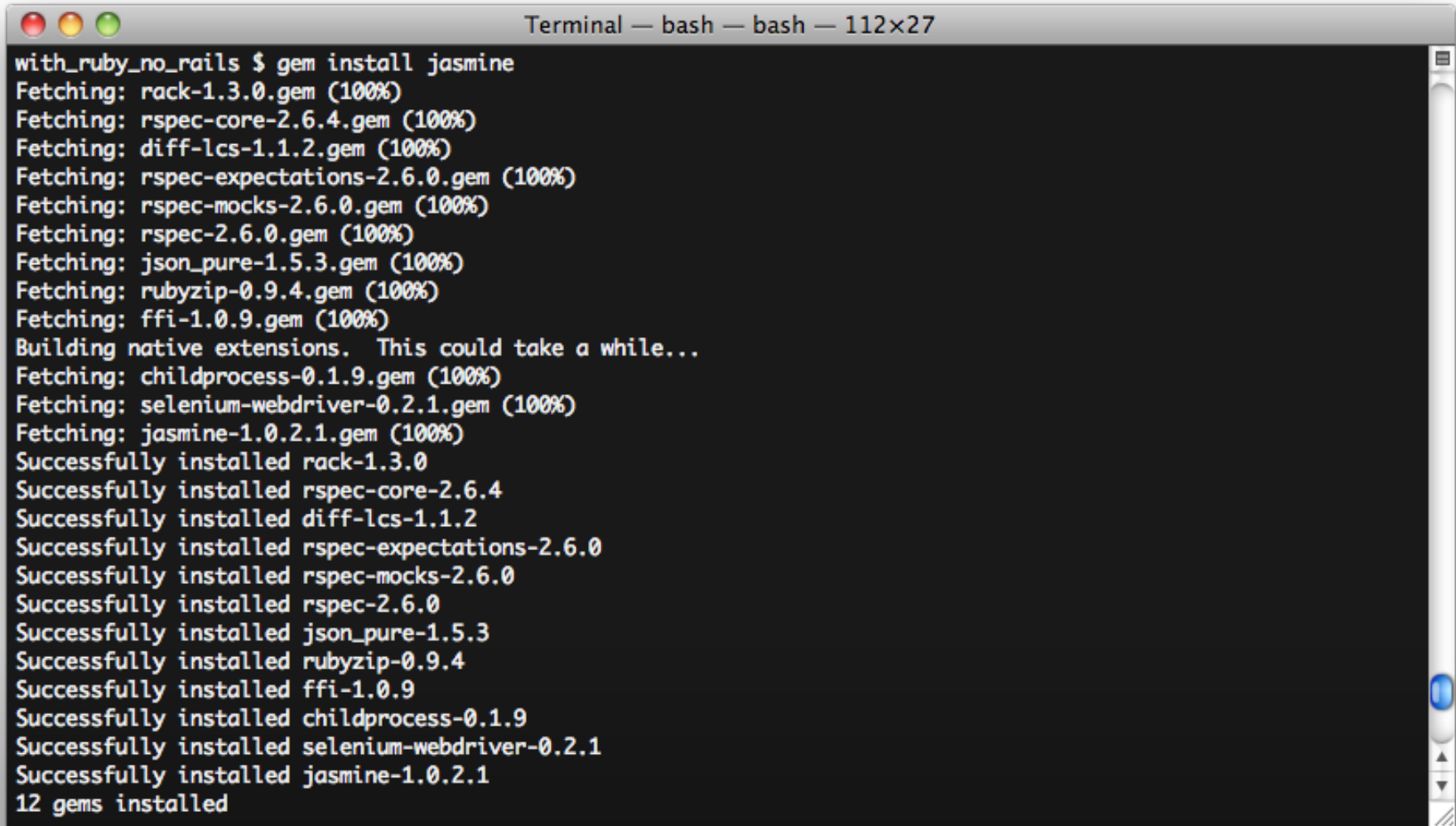
# Passed!



# Jasmine with Ruby(no Rails)

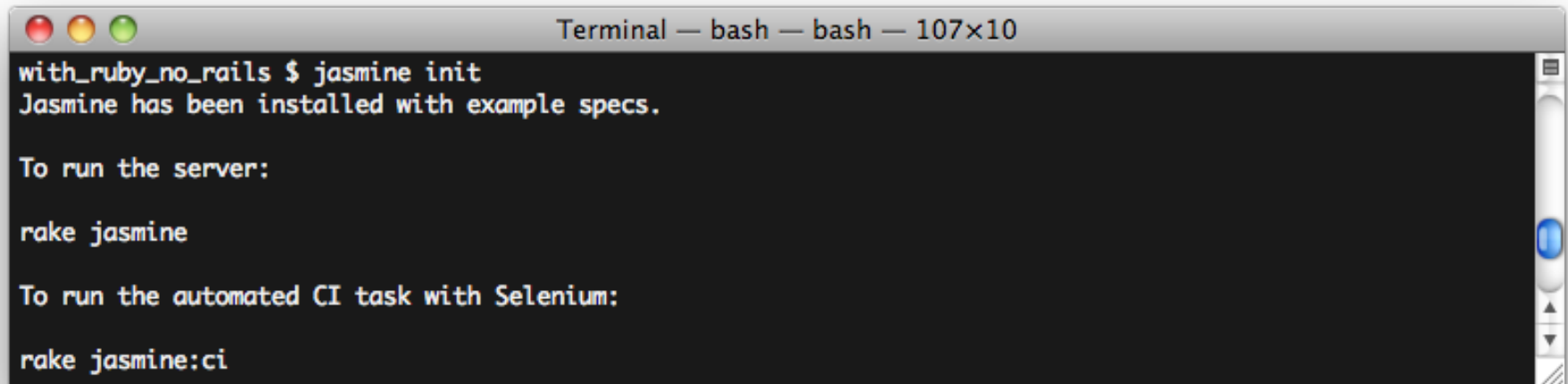


# \$ gem install jasmine

A screenshot of a macOS Terminal window. The title bar at the top reads "Terminal — bash — bash — 112x27". The terminal content shows the command "with\_ruby\_no\_rails \$ gem install jasmine" being executed. It lists the fetching of 12 gems, all at 100% completion. A message "Building native extensions. This could take a while..." appears. Finally, it lists the successful installation of each gem, ending with "12 gems installed".

```
with_ruby_no_rails $ gem install jasmine
Fetching: rack-1.3.0.gem (100%)
Fetching: rspec-core-2.6.4.gem (100%)
Fetching: diff-lcs-1.1.2.gem (100%)
Fetching: rspec-expectations-2.6.0.gem (100%)
Fetching: rspec-mocks-2.6.0.gem (100%)
Fetching: rspec-2.6.0.gem (100%)
Fetching: json_pure-1.5.3.gem (100%)
Fetching: rubyzip-0.9.4.gem (100%)
Fetching: ffi-1.0.9.gem (100%)
Building native extensions. This could take a while...
Fetching: childprocess-0.1.9.gem (100%)
Fetching: selenium-webdriver-0.2.1.gem (100%)
Fetching: jasmine-1.0.2.1.gem (100%)
Successfully installed rack-1.3.0
Successfully installed rspec-core-2.6.4
Successfully installed diff-lcs-1.1.2
Successfully installed rspec-expectations-2.6.0
Successfully installed rspec-mocks-2.6.0
Successfully installed rspec-2.6.0
Successfully installed json_pure-1.5.3
Successfully installed rubyzip-0.9.4
Successfully installed ffi-1.0.9
Successfully installed childprocess-0.1.9
Successfully installed selenium-webdriver-0.2.1
Successfully installed jasmine-1.0.2.1
12 gems installed
```

# \$ jasmine init

A screenshot of a macOS Terminal window. The title bar at the top reads "Terminal — bash — bash — 107x10". The terminal content shows the command "with\_ruby\_no\_rails \$ jasmine init" being executed. The output is "Jasmine has been installed with example specs." followed by instructions: "To run the server:" and the command "rake jasmine", and "To run the automated CI task with Selenium:" and the command "rake jasmine:ci".

```
Terminal — bash — bash — 107x10
with_ruby_no_rails $ jasmine init
Jasmine has been installed with example specs.

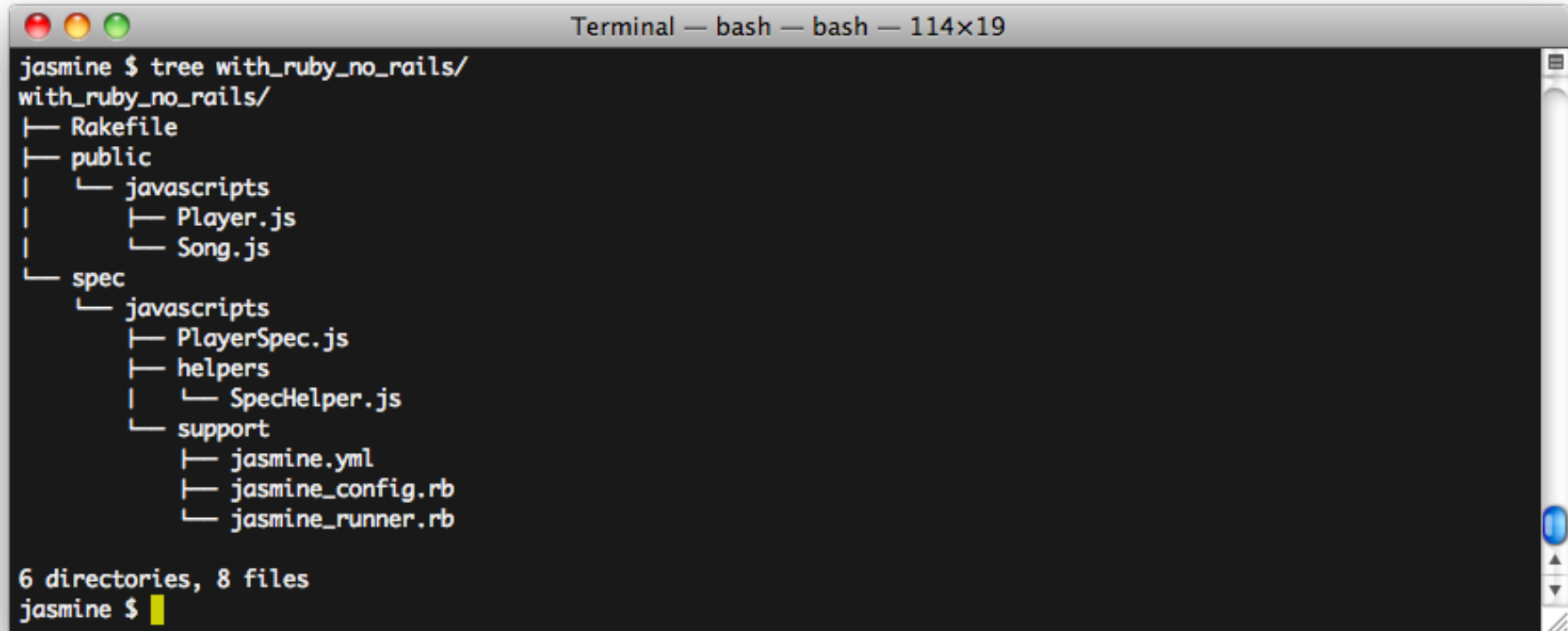
To run the server:

rake jasmine

To run the automated CI task with Selenium:

rake jasmine:ci
```

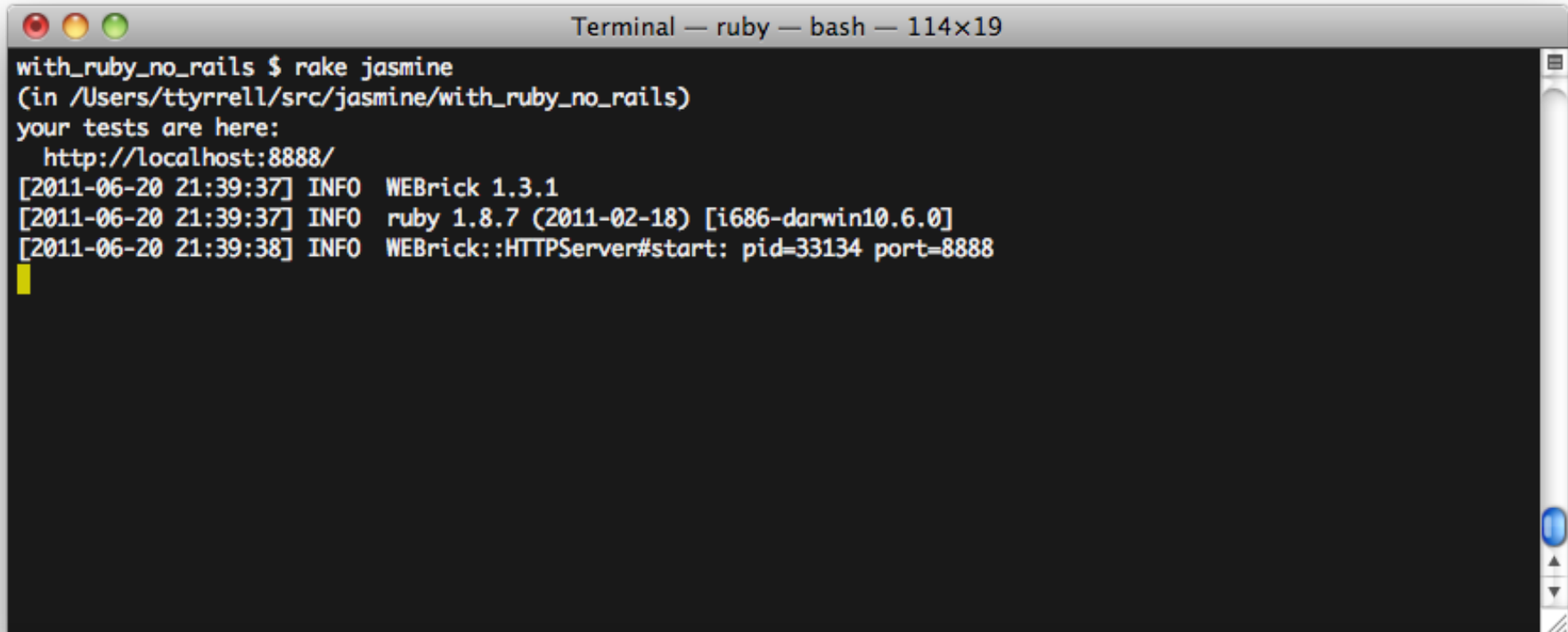
# Generated folder structure



```
Terminal — bash — bash — 114x19
jasmine $ tree with_ruby_no_rails/
with_ruby_no_rails/
├── Rakefile
├── public
│   ├── javascripts
│   │   ├── Player.js
│   │   └── Song.js
└── spec
    ├── javascripts
    │   ├── PlayerSpec.js
    │   ├── helpers
    │   │   └── SpecHelper.js
    └── support
        ├── jasmine.yml
        ├── jasmine_config.rb
        └── jasmine_runner.rb

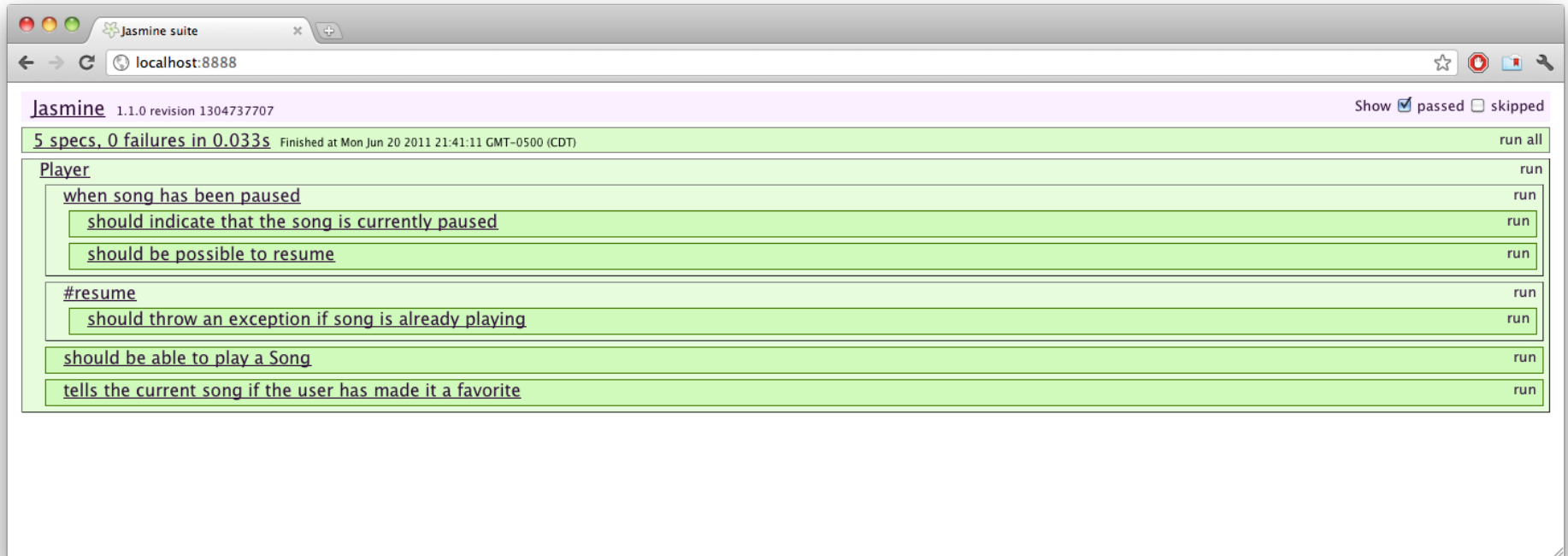
6 directories, 8 files
jasmine $
```

# \$ rake jasmine

A screenshot of a macOS Terminal window. The title bar at the top reads "Terminal — ruby — bash — 114x19". The terminal content shows the command "rake jasmine" being executed in a directory named "with\_ruby\_no\_rails". It displays the URL "http://localhost:8888/" and three log messages from WEBrick: the version (1.3.1), the Ruby version (1.8.7), and the server start message (pid=33134 port=8888). A yellow cursor is visible on the line following the last log message.

```
with_ruby_no_rails $ rake jasmine
(in /Users/ttyrrell/src/jasmine/with_ruby_no_rails)
your tests are here:
  http://localhost:8888/
[2011-06-20 21:39:37] INFO  WEBrick 1.3.1
[2011-06-20 21:39:37] INFO  ruby 1.8.7 (2011-02-18) [i686-darwin10.6.0]
[2011-06-20 21:39:38] INFO  WEBrick::HTTPServer#start: pid=33134 port=8888
```

# http://localhost:8888



The screenshot shows a web browser window with the title "Jasmine suite" and the address bar displaying "localhost:8888". The main content area shows the results of a Jasmine test suite. At the top, a purple header bar displays "jasmine 1.1.0 revision 1304737707" and a "Show" button with a checked "passed" checkbox and an unchecked "skipped" checkbox. Below this, a green bar indicates "5 specs, 0 failures in 0.033s" and "Finished at Mon Jun 20 2011 21:41:11 GMT-0500 (CDT)". The test results are organized into a table with a "run" button for each spec.

Spec Name	Action
<b>Player</b>	run
when song has been paused	run
should indicate that the song is currently paused	run
should be possible to resume	run
#resume	run
should throw an exception if song is already playing	run
should be able to play a Song	run
tells the current song if the user has made it a favorite	run

# \$ rake jasmine:ci

A screenshot of a macOS Terminal window. The title bar at the top reads "Terminal — bash — bash — 112x15". The terminal content shows the execution of the command "rake jasmine:ci" within a directory named "with\_ruby\_no\_rails". The output includes several informational messages from WEBrick and Ruby, indicating that the jasmine server started on port 62082. After a brief wait, the test suite completed successfully in 1.07 seconds, with 5 examples and 0 failures. The prompt "with\_ruby\_no\_rails \$" is visible at the bottom.

```
with_ruby_no_rails $ rake jasmine:ci
(in /Users/ttyrrell/src/jasmine/with_ruby_no_rails)
/Users/ttyrrell/.rvm/rubies/ruby-1.8.7-p334/bin/ruby -S rspec --colour --format progress spec/javascripts/support/jasmine_runner.rb
[2011-06-20 21:50:37] INFO WEBrick 1.3.1
[2011-06-20 21:50:37] INFO ruby 1.8.7 (2011-02-18) [i686-darwin10.6.0]
[2011-06-20 21:50:39] INFO WEBrick::HTTPServer#start: pid=33369 port=62082
Waiting for jasmine server on 62082...
jasmine server started.
Waiting for suite to finish in browser ...
.....

Finished in 1.07 seconds
5 examples, 0 failures
with_ruby_no_rails $
```



# Jasmine with JQuery



# jasmine-jquery Matchers

- \* toBe(jQuerySelector)
- \* toBeChecked()
- \* toBeEmpty()
- \* toBeHidden()
- \* toBeSelected()
- \* toBeVisible()
- \* toContain(jQuerySelector)
- \* toExist()
- \* toHaveAttr(attributeName, attributeValue)
- \* toHaveBeenTriggeredOn(selector)
- \* toHaveClass(className)
- \* toHaveData(key, value)
- \* toHaveHtml(string)
- \* toHaveId(id)
- \* toHaveText(string)
- \* toHaveValue(value)
- \* toBeDisabled()



"A lightweight, easy to use Javascript `<span>` injector for radical Web Typography"

@davatron5000



# Lettering.js

```
<h1 class="fancy_title">Some Title</h1>
```

```
<script>  
  $(document).ready(function() {  
    $(".fancy_title").lettering();  
  });  
</script>
```

# Lettering.js results

```
<h1 class="fancy_title">  
  <span class="char1">S</span>  
  <span class="char2">o</span>  
  <span class="char3">m</span>  
  <span class="char4">e</span>  
  <span class="char5"></span>  
  <span class="char6">T</span>  
  <span class="char7">i</span>  
  <span class="char8">t</span>  
  <span class="char9">l</span>  
  <span class="char10">e</span>  
</h1>
```

# Standalone folder structure + more

```
Terminal — bash — bash — 69x20
jasmine_with_jquery $ tree .
.
├── SpecRunner.html
├── lib
│   ├── jasmine-1.0.2
│   │   ├── MIT.LICENSE
│   │   ├── jasmine-html.js
│   │   ├── jasmine.css
│   │   └── jasmine.js
│   ├── jasmine-jquery-1.2.0.js
│   ├── jquery-1.6.1.min.js
│   └── jquery.lettering-0.6.1.min.js
└── spec
    ├── LetteringSpec.js
    └── javascripts
        └── fixtures
            └── myfixture.html

5 directories, 10 files
jasmine_with_jquery $
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <title>Jasmine Test Runner</title>
  <link rel="stylesheet" type="text/css" href="lib/jasmine-1.0.2/jasmine.css">
  <script type="text/javascript" src="lib/jasmine-1.0.2/jasmine.js"></script>
  <script type="text/javascript" src="lib/jasmine-1.0.2/jasmine-html.js"></script>

  <script type="text/javascript" src="lib/jquery-1.6.1.min.js"></script>
  <script type="text/javascript" src="lib/jasmine-jquery-1.2.0.js"></script>
  <script type="text/javascript" src="lib/jquery.lettering-0.6.1.min.js"></script>

  <script type="text/javascript" src="spec/LetteringSpec.js"></script>

</head>
<body>

<script type="text/javascript">
  jasmine.getEnv().addReporter(new jasmine.TrivialReporter());
  jasmine.getEnv().execute();
</script>

</body>
</html>
```



```
describe("Lettering", function(){  
  beforeEach(function(){  
    //loadFixtures('myfixture.html');  
    setFixtures('<h1 class="fancy_title">Some Title</h1>');  
    $('.fancy_title').lettering();  
  });
```

```
  it("should be visible", function() {  
    expect($('.fancy_title')).toBeVisible();  
  });
```

```
  it("should have 10 spans", function(){  
    expect($('.fancy_title > span').length).toEqual(10);  
  });
```

```
  it("should contain 'S' as the text of the first span", function(){  
    expect($('.fancy_title > span:first').text()).toEqual('S');  
  });
```

```
  it("should have a class of 'char1' on the first span", function(){  
    expect($('.fancy_title > span:first')).toHaveClass('char1');  
  });  
});
```

```
describe("Lettering", function(){
  beforeEach(function(){
    //loadFixtures('myfixture.html');
    setFixtures('<h1 class="fancy_title">Some Title</h1>');
    $('fancy_title').lettering();
  });
```

```
it("should be visible", function() {
  expect($('fancy_title')).toBeVisible();
});
```

```
it("should have 10 spans", function(){
  expect($('fancy_title > span').length).toEqual(10);
});
```

```
it("should contain 'S' as the text of the first span", function(){
  expect($('fancy_title > span:first').text()).toEqual('S');
});
```

```
it("should have a class of 'char1' on the first span", function(){
  expect($('fancy_title > span:first')).toHaveClass('char1');
});
});
```

```
describe("Lettering", function(){
  beforeEach(function(){
    //loadFixtures('myfixture.html');
    setFixtures('<h1 class="fancy_title">Some Title</h1>');
    $('.fancy_title').lettering();
  });
```

```
it("should be visible", function() {
  expect($('.fancy_title')).toBeVisible();
});
```

```
it("should have 10 spans", function(){
  expect($('.fancy_title > span').length).toEqual(10);
});
```

```
it("should contain 'S' as the text of the first span", function(){
  expect($('.fancy_title > span:first').text()).toEqual('S');
});
```

```
it("should have a class of 'char1' on the first span", function(){
  expect($('.fancy_title > span:first')).toHaveClass('char1');
});
});
```

```
describe("Lettering", function(){
  beforeEach(function(){
    //loadFixtures('myfixture.html');
    setFixtures('<h1 class="fancy_title">Some Title</h1>');
    $('.fancy_title').lettering();
  });
```

```
it("should be visible", function() {
  expect($('.fancy_title')).toBeVisible();
});
```

```
it("should have 10 spans", function(){
  expect($('.fancy_title > span').length).toEqual(10);
});
```

```
it("should contain 'S' as the text of the first span", function(){
  expect($('.fancy_title > span:first').text()).toEqual('S');
});
```

```
it("should have a class of 'char1' on the first span", function(){
  expect($('.fancy_title > span:first')).toHaveClass('char1');
});
});
```

```
describe("Lettering", function(){
  beforeEach(function(){
    //loadFixtures('myfixture.html');
    setFixtures('<h1 class="fancy_title">Some Title</h1>');
    $('.fancy_title').lettering();
  });
```

```
it("should be visible", function() {
  expect($('.fancy_title')).toBeVisible();
});
```

```
it("should have 10 spans", function(){
  expect($('.fancy_title > span').length).toEqual(10);
});
```

```
it("should contain 'S' as the text of the first span", function(){
  expect($('.fancy_title > span:first').text()).toEqual('S');
});
```

```
it("should have a class of 'char1' on the first span", function(){
  expect($('.fancy_title > span:first')).toHaveClass('char1');
});
});
```

# Jasmine with Rails

```
gem "jasmine"
```

```
$ bundle install
```

```
$ jasmine init (optional)
```

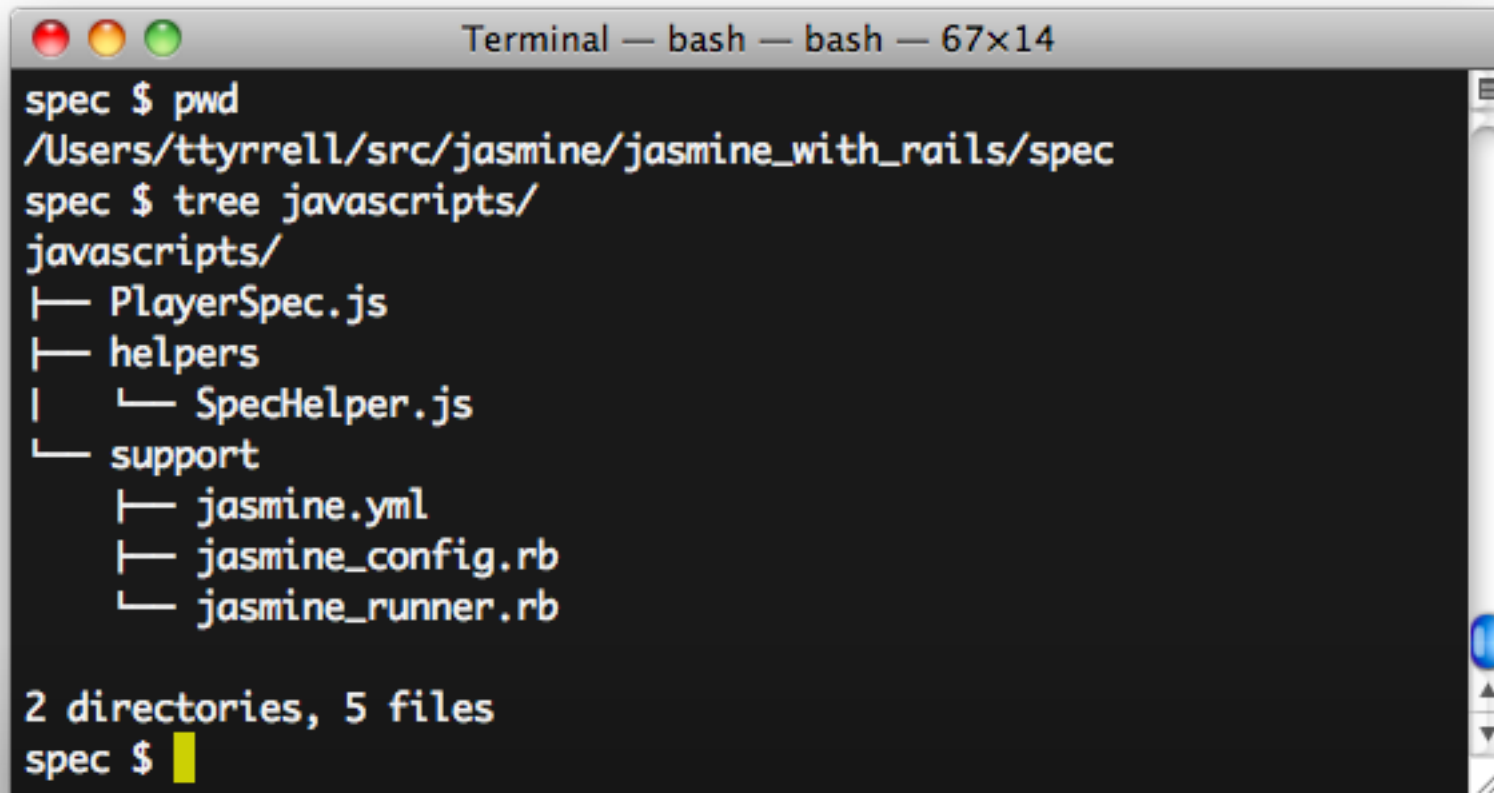
```
$ rake jasmine
```

or

```
$ rake jasmine:ci
```

```
http://localhost:8888
```

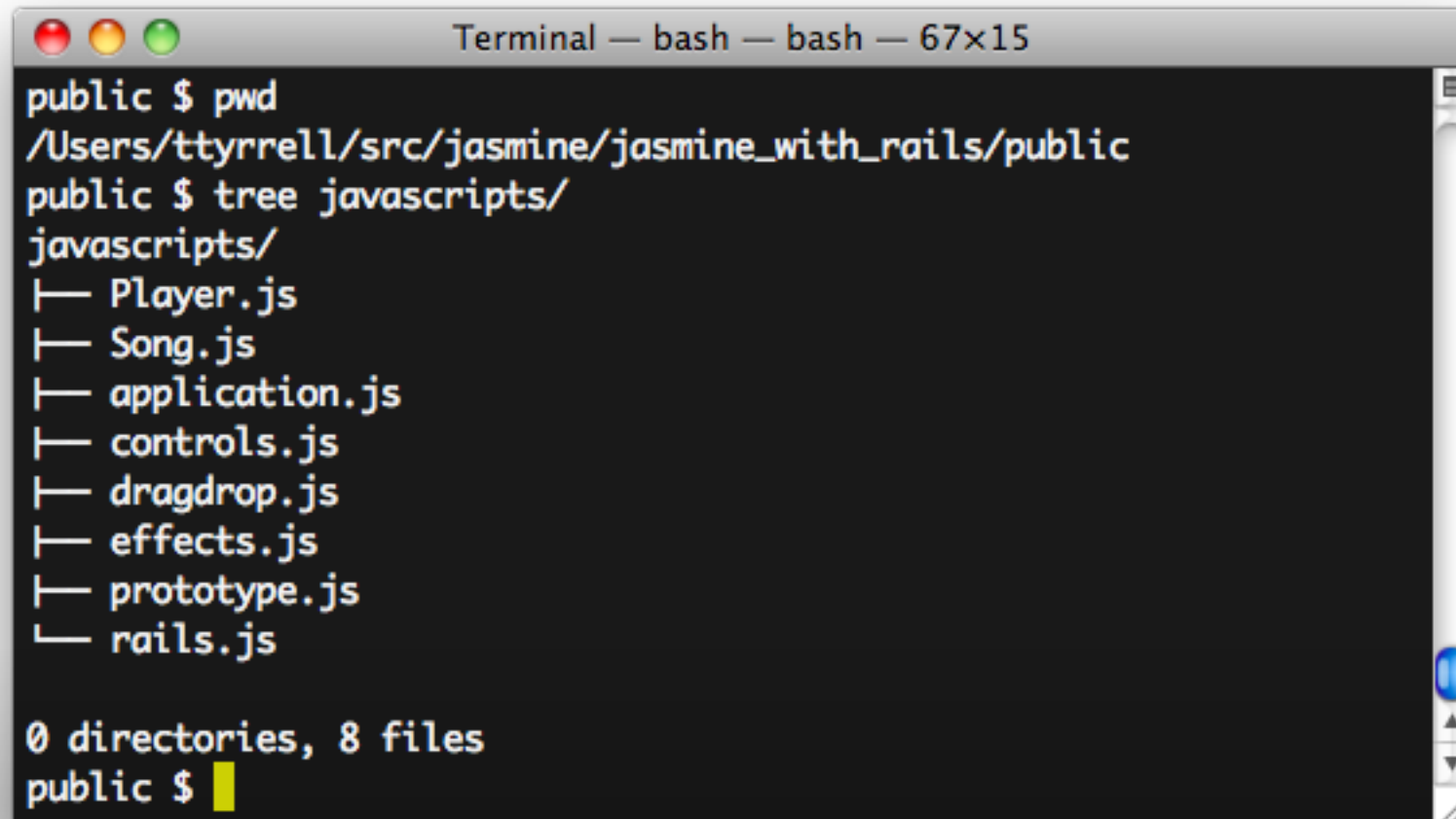
# Generated Rails spec folder

A screenshot of a macOS Terminal window. The title bar at the top reads "Terminal — bash — bash — 67x14". The terminal has a black background with white text. The user has entered the command "spec \$ pwd" and the output is "/Users/ttyrrell/src/jasmine/jasmine\_with\_rails/spec". Then, the user entered "spec \$ tree javascripts/" and the output shows a directory tree for "javascripts/". The tree includes a "PlayerSpec.js" file, a "helpers" directory containing "SpecHelper.js", and a "support" directory containing "jasmine.yml", "jasmine\_config.rb", and "jasmine\_runner.rb". At the bottom, it says "2 directories, 5 files" and the prompt "spec \$" is followed by a yellow cursor.

```
spec $ pwd
/Users/ttyrrell/src/jasmine/jasmine_with_rails/spec
spec $ tree javascripts/
javascripts/
├── PlayerSpec.js
├── helpers
│   └── SpecHelper.js
└── support
    ├── jasmine.yml
    ├── jasmine_config.rb
    └── jasmine_runner.rb

2 directories, 5 files
spec $
```

# Generated default javascript objects

A screenshot of a macOS Terminal window. The title bar reads "Terminal — bash — bash — 67x15". The terminal shows the current directory as "/Users/ttyrrell/src/jasmine/jasmine\_with\_rails/public" and a tree view of the "javascripts/" directory containing eight files: Player.js, Song.js, application.js, controls.js, dragdrop.js, effects.js, prototype.js, and rails.js. A summary line indicates "0 directories, 8 files".

```
public $ pwd
/Users/ttyrrell/src/jasmine/jasmine_with_rails/public
public $ tree javascripts/
javascripts/
├── Player.js
├── Song.js
├── application.js
├── controls.js
├── dragdrop.js
├── effects.js
├── prototype.js
└── rails.js

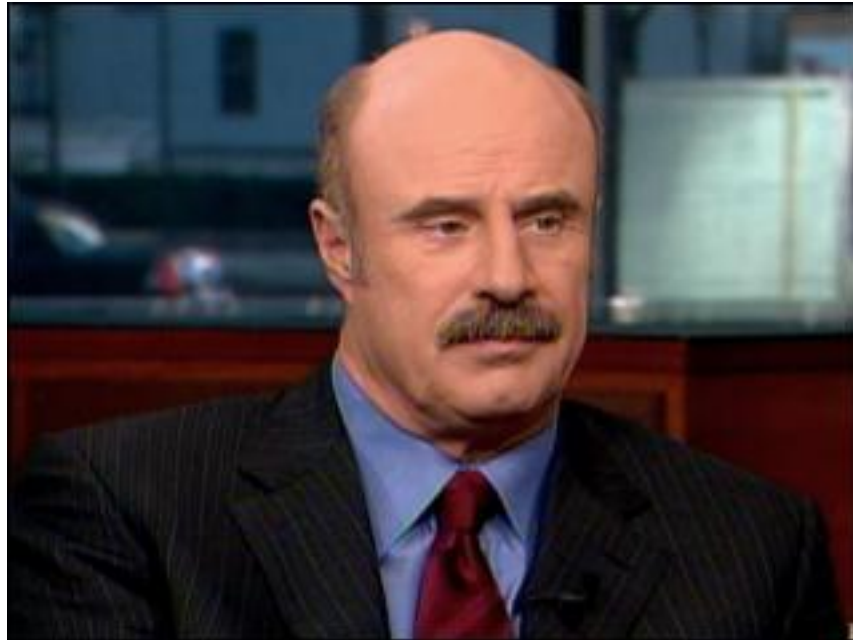
0 directories, 8 files
public $
```



# \$ rake jasmine

The screenshot shows a web browser window with the title "Jasmine suite" and the address bar displaying "localhost:8888". The main content area shows the results of a Jasmine test suite. At the top, it says "Jasmine 1.1.0 revision 1304737707" and "Show ☒ passed ☐ skipped". Below this, a green bar indicates "5 specs, 0 failures in 0.015s" and "Finished at Tue Jun 21 2011 00:37:35 GMT-0500 (CDT)". The test suite is titled "Player" and contains several specs, each with a "run" button next to it. The specs are: "when song has been paused" (which contains two sub-specs: "should indicate that the song is currently paused" and "should be possible to resume"), "#resume" (with sub-spec "should throw an exception if song is already playing"), "should be able to play a Song", and "tells the current song if the user has made it a favorite".

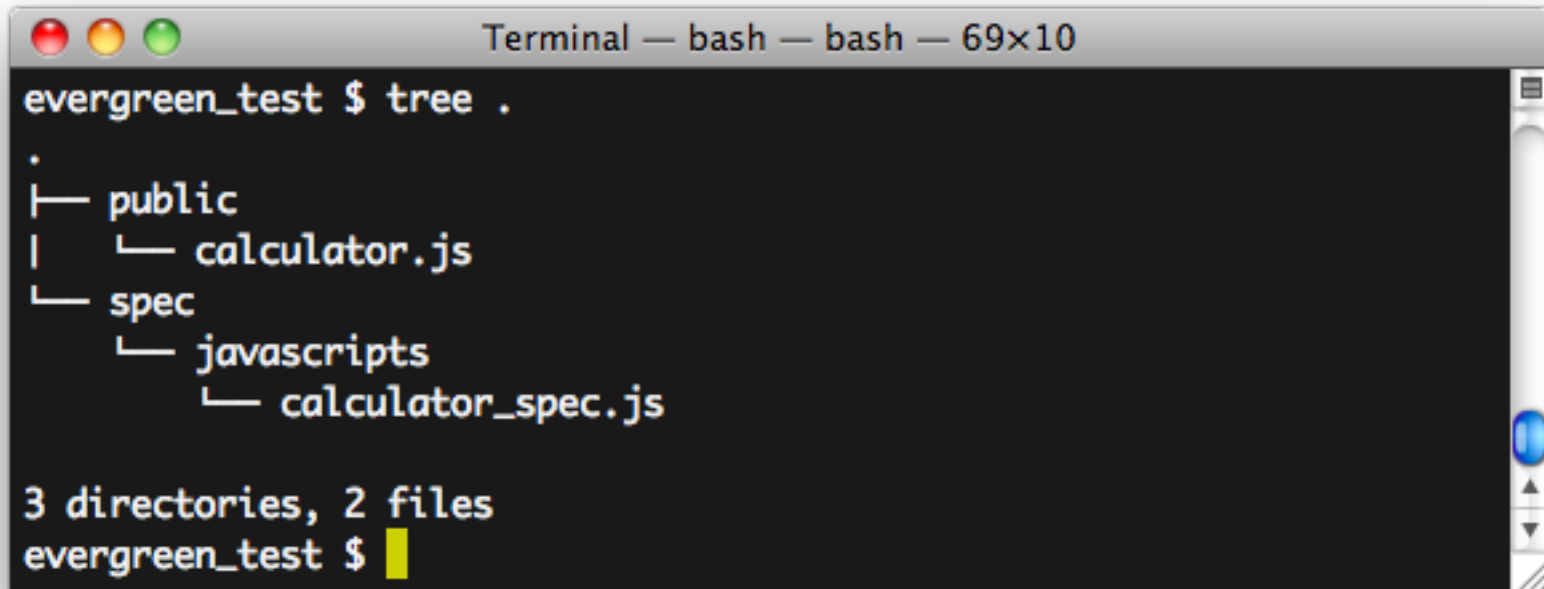
Spec	Status
Player	run
when song has been paused	run
should indicate that the song is currently paused	run
should be possible to resume	run
#resume	run
should throw an exception if song is already playing	run
should be able to play a Song	run
tells the current song if the user has made it a favorite	run



\$ gem install evergreen

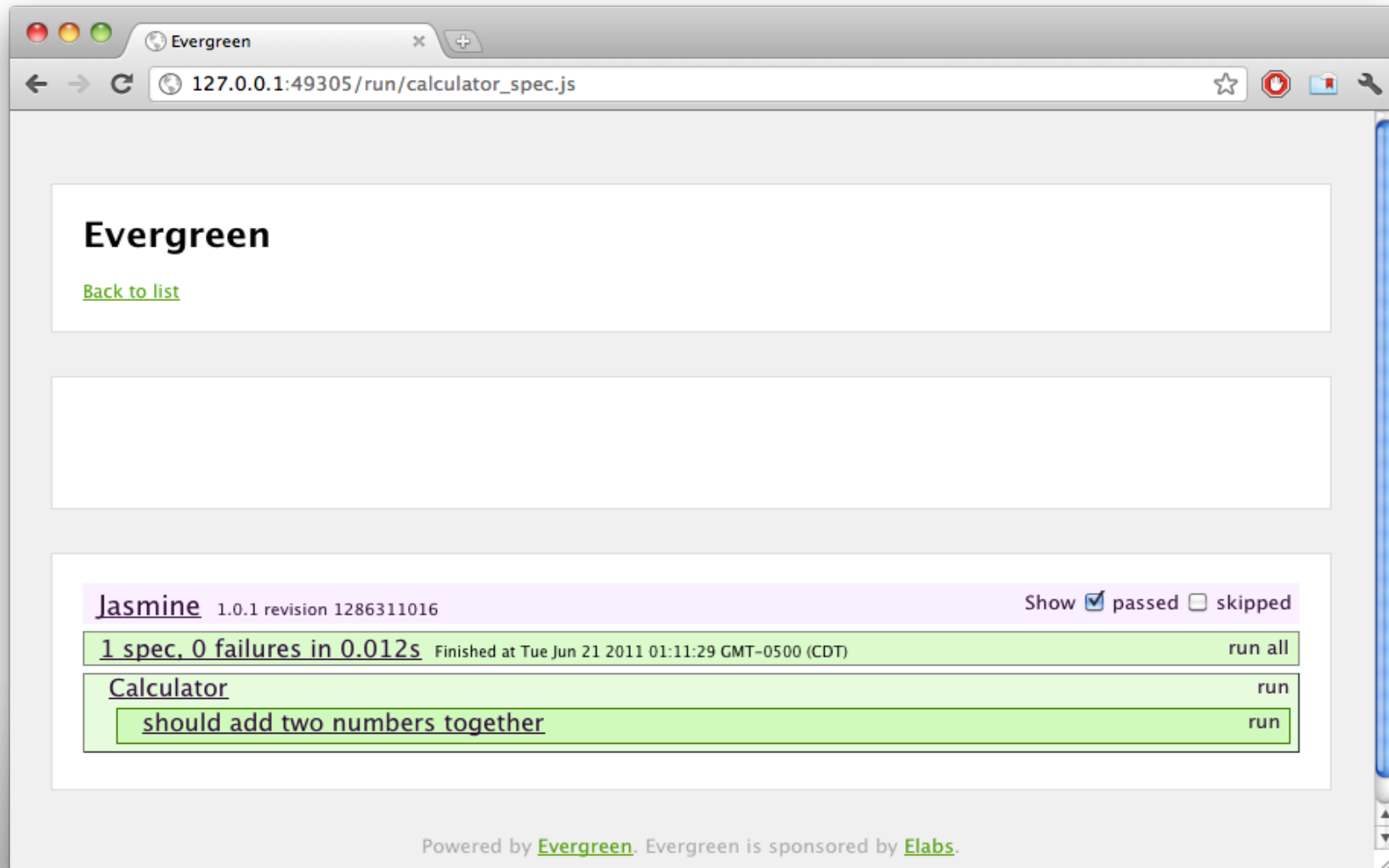


# Evergreen default folder paths (not generated)

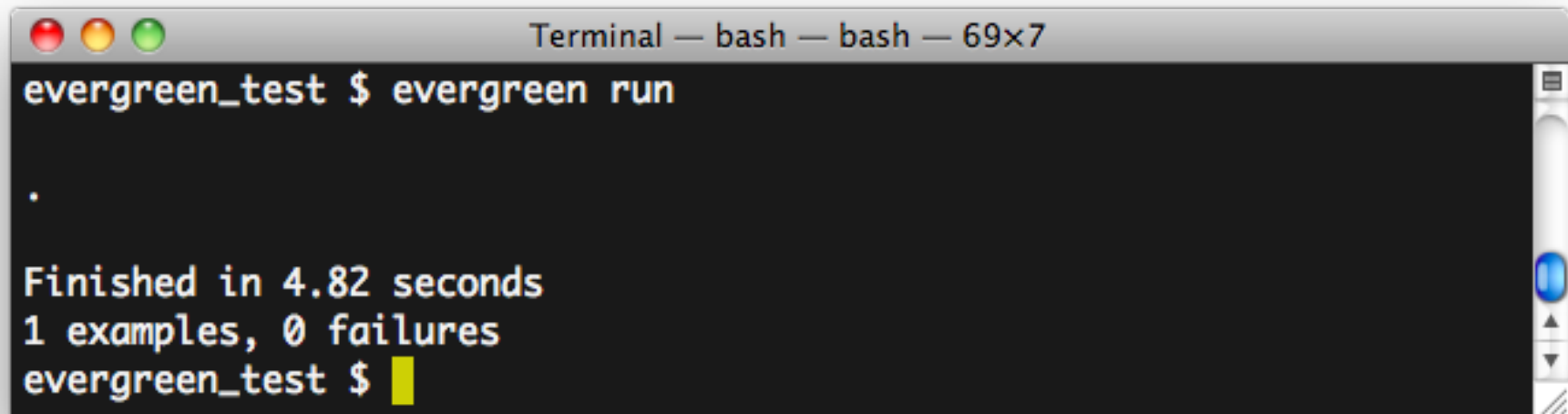
A screenshot of a macOS Terminal window. The title bar at the top reads "Terminal — bash — bash — 69x10". The terminal content shows the command "evergreen\_test \$ tree ." and its output, which is a directory tree. The tree shows a root directory with two subdirectories: "public" and "spec". The "public" directory contains a file named "calculator.js". The "spec" directory contains a subdirectory named "javascripts", which in turn contains a file named "calculator\_spec.js". Below the tree, the summary "3 directories, 2 files" is displayed. The prompt "evergreen\_test \$" is followed by a yellow cursor.

```
Terminal — bash — bash — 69x10
evergreen_test $ tree .
.
├── public
│   └── calculator.js
└── spec
    ├── javascripts
    │   └── calculator_spec.js
    └──
3 directories, 2 files
evergreen_test $
```

# \$ evergreen serve



# \$ evergreen run

A screenshot of a macOS Terminal window. The title bar at the top reads "Terminal — bash — bash — 69x7". The terminal has a black background with white text. The prompt "evergreen\_test \$" is followed by the command "evergreen run". The output shows a single dot "." on the next line, followed by "Finished in 4.82 seconds" and "1 examples, 0 failures". The prompt "evergreen\_test \$" appears again with a yellow cursor block.

```
Terminal — bash — bash — 69x7
evergreen_test $ evergreen run

.

Finished in 4.82 seconds
1 examples, 0 failures
evergreen_test $
```

# Evergreen and Rails 3

```
gem 'evergreen', :require =>  
'evergreen/rails'localhost:  
3000/evergreenrake spec:  
javascripts
```

# CoffeeScript???





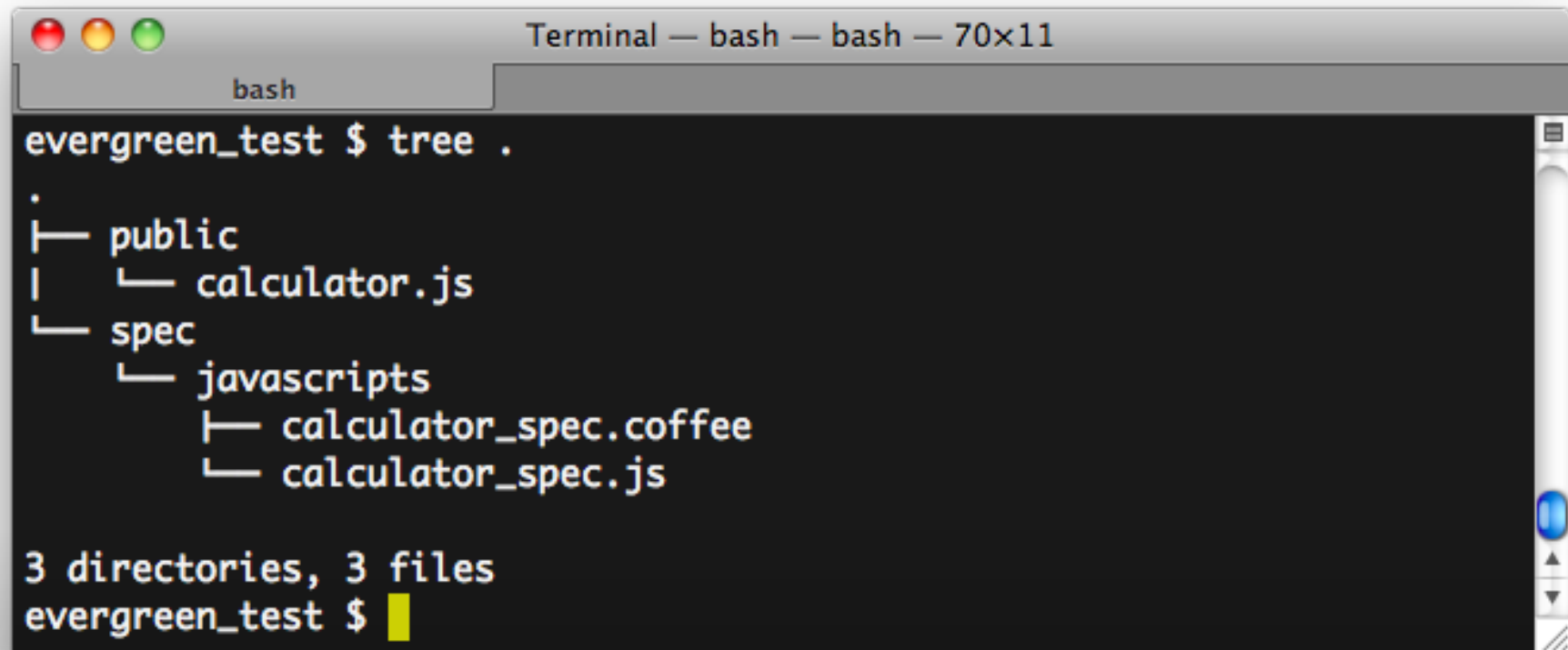
# Installing CoffeeScript

brew install node.js (then add to PATH, NODE\_PATH)

curl <http://npmjs.org/install.sh> | sh

npm install -g coffee-script

# Add a "\_spec.coffee" file

A screenshot of a macOS Terminal window. The title bar at the top reads "Terminal — bash — bash — 70x11". Below the title bar, there is a tab labeled "bash". The terminal content shows the command "evergreen\_test \$ tree ." and its output, which is a tree view of the current directory. The tree shows a "public" directory, a "spec" directory, and a "javascripts" directory. The "spec" directory contains "calculator\_spec.coffee" and "calculator\_spec.js". The "javascripts" directory contains "calculator\_spec.coffee" and "calculator\_spec.js". At the bottom of the terminal, it says "3 directories, 3 files" and "evergreen\_test \$" followed by a yellow cursor.

```
evergreen_test $ tree .  
.  
├── public  
│   └── calculator.js  
└── spec  
    ├── javascripts  
    │   ├── calculator_spec.coffee  
    │   └── calculator_spec.js  
    └── calculator_spec.coffee  
    └── calculator_spec.js  
  
3 directories, 3 files  
evergreen_test $
```

```
require('/calculator.js')
```

describe 'Calculator', ->

beforeEach ->

@calc = new Calculator

it 'should add two numbers together', ->

expect(@calc.Add(1,1)).toEqual(2)

```
require('/calculator.js')
```

```
describe 'Calculator', ->
```

```
  beforeEach ->
```

```
    @calc = new Calculator
```

```
    it 'should add two numbers together', ->
```

```
      expect(@calc.Add(1,1)).toEqual(2)
```

```
require('/calculator.js')
```

```
describe 'Calculator', ->
```

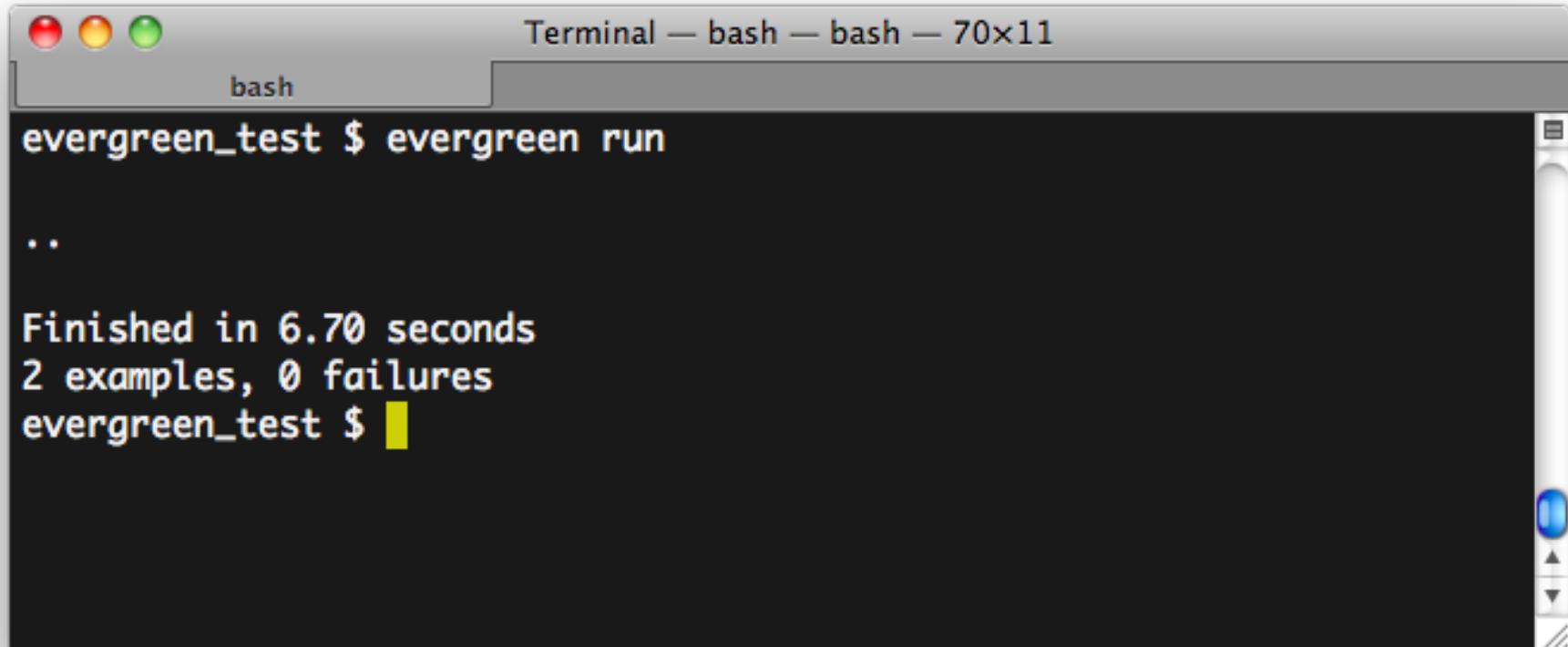
```
  beforeEach ->
```

```
    @calc = new Calculator
```

```
    it 'should add two numbers together', ->
```

```
      expect(@calc.Add(1,1)).toEqual(2)
```

# #winning

A screenshot of a macOS Terminal window. The title bar at the top reads "Terminal — bash — bash — 70x11". Below the title bar, there is a tab labeled "bash". The terminal content shows a command prompt "evergreen\_test \$" followed by the command "evergreen run". The output consists of two dots ".." on the next line, followed by the text "Finished in 6.70 seconds" and "2 examples, 0 failures". The prompt "evergreen\_test \$" is shown again with a yellow cursor. On the right side of the terminal window, there is a vertical scrollbar and a blue button at the bottom.

```
Terminal — bash — bash — 70x11
bash
evergreen_test $ evergreen run
..
Finished in 6.70 seconds
2 examples, 0 failures
evergreen_test $
```



# Node.js + Jasmine

"jasmine-node"

```
$ node specs.js
Started
..

Spec When making a regular customer preferred
Finished in 0 seconds
1 test, 0 assertions, 0 failures
```



Any other cool utilities?

# Summer Breeze

"Summer Breeze is a tool for generating fixture DOM output for Jasmine JavaScript tests directly from your Rails Views with a simple DSL for specifying fixtures"

[https://github.com/noelrappin/summer\\_breeze](https://github.com/noelrappin/summer_breeze)

# Rosie

"Rosie is a factory for building JavaScript objects, mostly useful for setting up test data. It is inspired by [factory\\_girl](#)."

<https://github.com/bkeepers/rosie>

# http://try-jasmine.herokuapp.com

The screenshot shows a web browser window with the URL `try-jasmine.herokuapp.com/index.html`. The page has a header with a "try jasmine!" button and instructions: "mac: ⌘-return, windows: ctrl-enter". There are also buttons for "see CoffeeScript" and "clear saved specs", and a "Fork me on GitHub" banner.

The main content area is divided into two sections: "specs" and "results".

**specs**

macros: `describe` `it` `expect` `spyOn` `beforeEach`

```
1 describe 'Calculator', ->
2   beforeEach ->
3     @calc = new Calculator
4     it 'should add two numbers together', ->
5       expect(@calc.Add(1,1)).toEqual(2)
```

**results**

Jasmine 1.0.2 revision 1298837858 Show ☐ passed ☐ skipped

1 spec, 0 failures in 0.02s Finished at Tue Jun 21 2011 01:43:04 run all  
GMT-0500 (CDT)

**source**

```
1 function Calculator () {
2   this.Add = function(num1, num2){
3     return num1 + num1;
4   }
5 }
```

I am done.

Questions?

<http://speakerrate.com/talks/7840-testing-javascript-with-jasmine>

## References:

<http://pivotal.github.com/jasmine/>

<https://github.com/jnicklas/evergreen>

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

<http://obtiva.com/blog/112-javascript-specs-with-jasmine-a-primer-for-rubyists-part-1>

<http://obtiva.com/blog/119>

<http://daverupert.com/2010/09/lettering-js/>

<http://www.engineyard.com/university/screencasts/javascript-tests-with-jasmine>

<http://blog.levid.com/?p=29>

<http://blog.carbonfive.com/2010/10/21/rspec-best-practices/>