

# Jasmine

The first steps to *finally* test-driving your JavaScript

You can find this presentation & code at:

[http://v.gd/jasmine\\_intro](http://v.gd/jasmine_intro)



Who?



You!

What?



Jasmine!

Where?

Jasmine: BDD for Javascript | Jasmine

http://pivotal.github.com/jasmine/ Google

Jasmine: BDD for Javascript | Jasmine

Fork me on GitHub

# Jasmine

- [Welcome](#)
- [Download](#)
- [Release Notes](#)
- Documentation
  - [User Guide](#)
  - [Background](#)
  - [Suites and Specs](#)
  - [Matchers](#)
  - [Before and After](#)
  - [Spies](#)
  - [Asynchronous Specs](#)
- [Using the Jasmine Gem](#)
- [API Documentation](#)
- [Contributor Guide](#)
- [Related Projects](#)
- [Who's Using Jasmine](#)

## 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.

```
describe("Jasmine", function() {
  it("makes testing JavaScript awesome!", function() {
    expect(yourCode).toBeLotsBetter();
  });
});
```

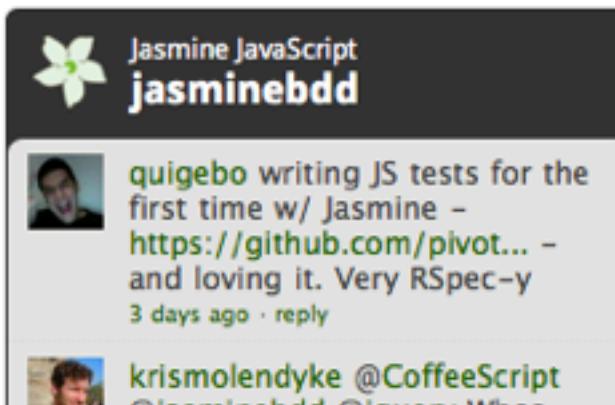
## Adding Jasmine to your Rails project

```
$ gem install jasmine
$ script/generate jasmine
$ rake spec
```

Jasmine can be run on a static web page, in your continuous integration environment, or with [node.js](#). See more in the documentation.

## Support

**Discussion:** [Google Group](#)  
**Group email:** [jasmine-js@googlegroups.com](mailto:jasmine-js@googlegroups.com)  
**Current Build Status:** [Jasmine at Pivotal Labs CI](#)  
**Report bugs at** [GitHub](#)  
**Project Backlog:** [Jasmine on Pivotal Tracker](#)



The screenshot shows a GitHub pull request interface. At the top, there's a header with the Jasmine logo and the text "Jasmine JavaScript jasminebdd". Below this, there are two commits listed:

- quigebo writing JS tests for the first time w/ Jasmine - <https://github.com/pivotal/jasmine/pull/12> - and loving it. Very RSpec-y 3 days ago · reply
- krismolendyke @CoffeeScript [@jasminebdd](https://github.com/krismolendyke) @jquery Whoa

When?



10

As soon as you're ready to jump in

Why?



Unfortunately, not enough time to cover the “why TDD my JavaScript?” question today

# More on Why:

<http://v.gd/javascript>



13

However, I did recently give a talk about why JavaScript is worthy of TDD.

iWork slides:

[http://public.iwork.com/document/?a=p94397438&d=JavaScript\\_Craftsmanship\\_38\\_TDD\\_Searls\\_9\\_47\\_13\\_47\\_2010.key](http://public.iwork.com/document/?a=p94397438&d=JavaScript_Craftsmanship_38_TDD_Searls_9_47_13_47_2010.key)

Slideshare:

<http://www.slideshare.net/searls/javascript-craftsmanship-why-javascript-is-worthy-of-tdd>

# How?

# I. Get Jasmine

Downloads | Jasmine

Downloads | Jasmine

 Jasmine

Fork me on GitHub

- [Welcome](#)
- [Download](#)
- [Release Notes](#)
- Documentation
  - [User Guide](#)
  - [Background](#)
  - [Suites and Specs](#)
  - [Matchers](#)
  - [Before and After](#)
  - [Spies](#)
  - [Asynchronous Specs](#)
- [Using the Jasmine Gem](#)
- [API Documentation](#)
- [Contributor Guide](#)
- [Related Projects](#)
- [Who's Using Jasmine](#)

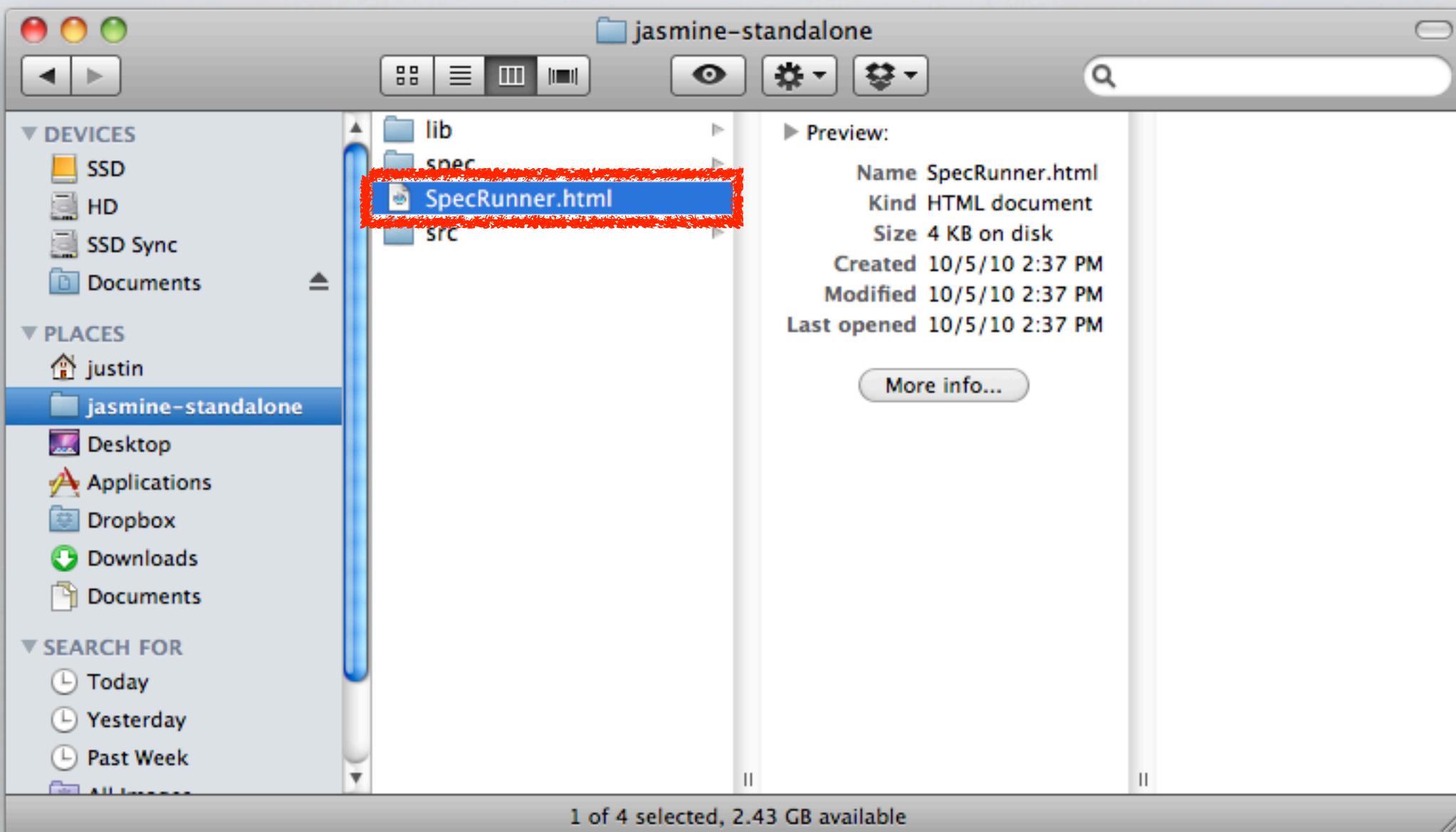
## Downloads

These files are for the standalone release, meant for inclusion on a static HTML page and manual management of files.

For a using Jasmine with Ruby on Rails, other Ruby frameworks, within continuous integration environments, or to gain more dynamic source and test file loading, see [Using the Jasmine Gem](#).

	Version	Size	Date	SHA1
<a href="#">jasmine-standalone-1.0.1.zip</a>	1.0.1	20k	2010/10/05 13:37:01 PDT	b2b3d00a7cb8d5cccd65a3356bb5ae15775328119
<a href="#">jasmine-standalone-1.0.0.zip</a>	1.0.0	20k	2010/09/14 12:54:38 PDT	1866f654a3ad0ab9109393ce31d6613c77916607
<a href="#">jasmine-standalone-1.0.0.rc1.zip</a>	1.0.0.rc1	19k	2010/08/26 13:09:41 PDT	20e7da22bc7ce3433331a5ad44eb199f4ff34065
<a href="#">jasmine-standalone-0.11.1.zip</a>	0.11.1	18k	2010/06/25 16:05:30 PDT	26998c7ca047e47f84c382a4efeb1dc5cb8661a6

For starters, download the latest standalone



Open `SpecRunner.html` in a browser

Jasmine Test Runner

file:///Volumes/Documents/Life/Speaking/jasmine-intro/jasmine-standalone/SpecRunner.html Google

Jasmine Test Runner

Jasmine 1.0.1 revision 1286311016

Show  passed  skipped

5 specs, 0 failures in 0.016s Finished at Fri Dec 31 2010 10:45:16 GMT-0500 (EST)

run all

Player

when song has been paused

should indicate that the song is currently paused

should be possible to resume

#resume

should throw an exception if song is already playing

should be able to play a Song

tells the current song if the user has made it a favorite

run

run

run

run

run

run

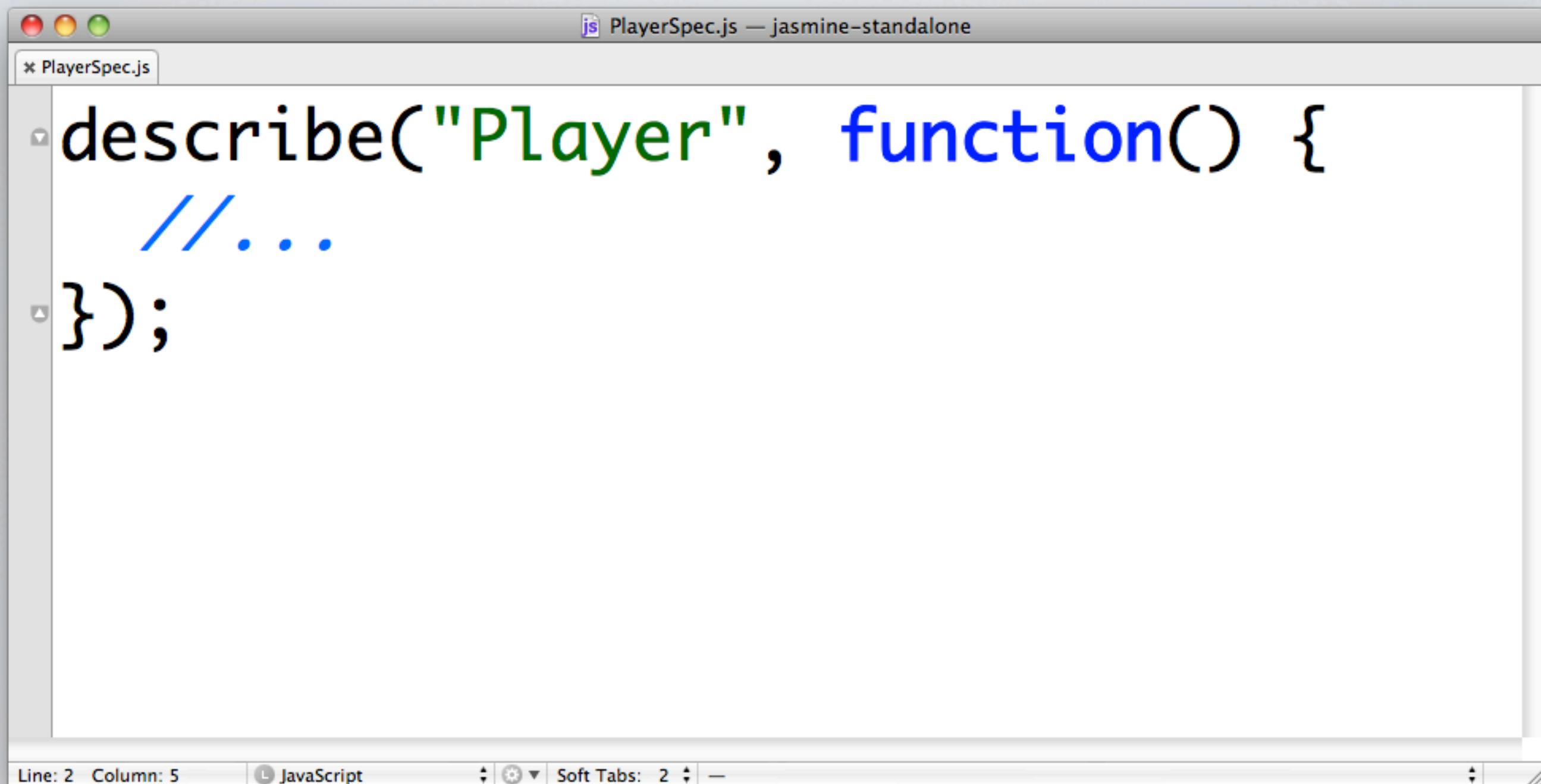
run

run

run

Check ‘passed’ to see the example specs

## 2. Start Spec'ing!



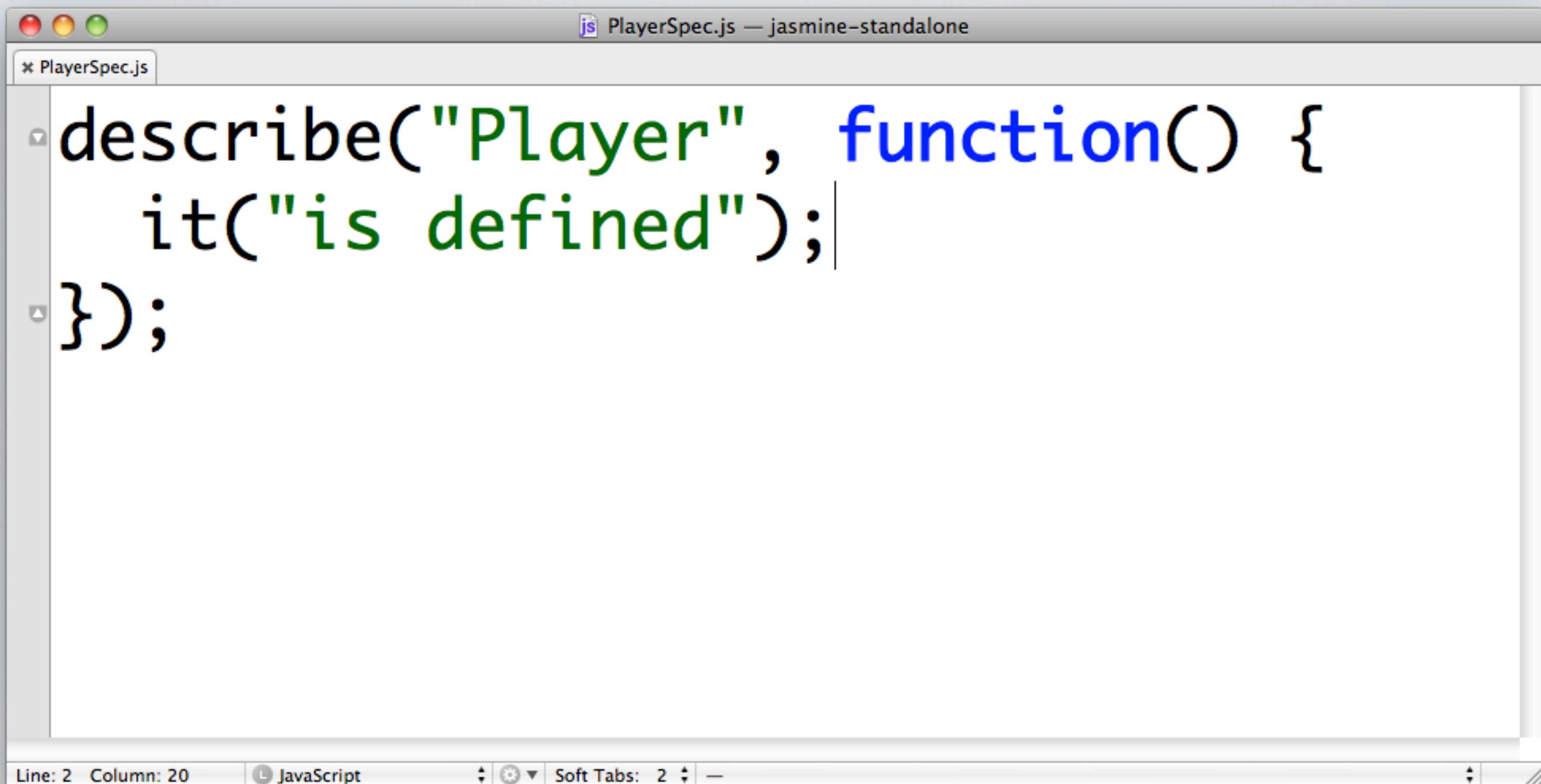
A screenshot of a code editor window titled "PlayerSpec.js — jasmine-standalone". The file contains the following JavaScript code:

```
describe("Player", function() {  
    //...  
});
```

The code editor interface includes a toolbar at the top with standard file operations, a status bar at the bottom showing "Line: 2 Column: 5", and a toolbar below it with icons for "JavaScript", "Soft Tabs: 2", and other settings.

## Let's start fresh!

This is a top-level example group; most spec files are arranged 1-to-1 with the code in the files they specify so they can be easily located and contain a single outer-most example group specified by invoking the `describe` function.



A screenshot of a code editor window titled "PlayerSpec.js — jasmine-standalone". The file contains the following JavaScript code:

```
describe("Player", function() {
  it("is defined");
});
```

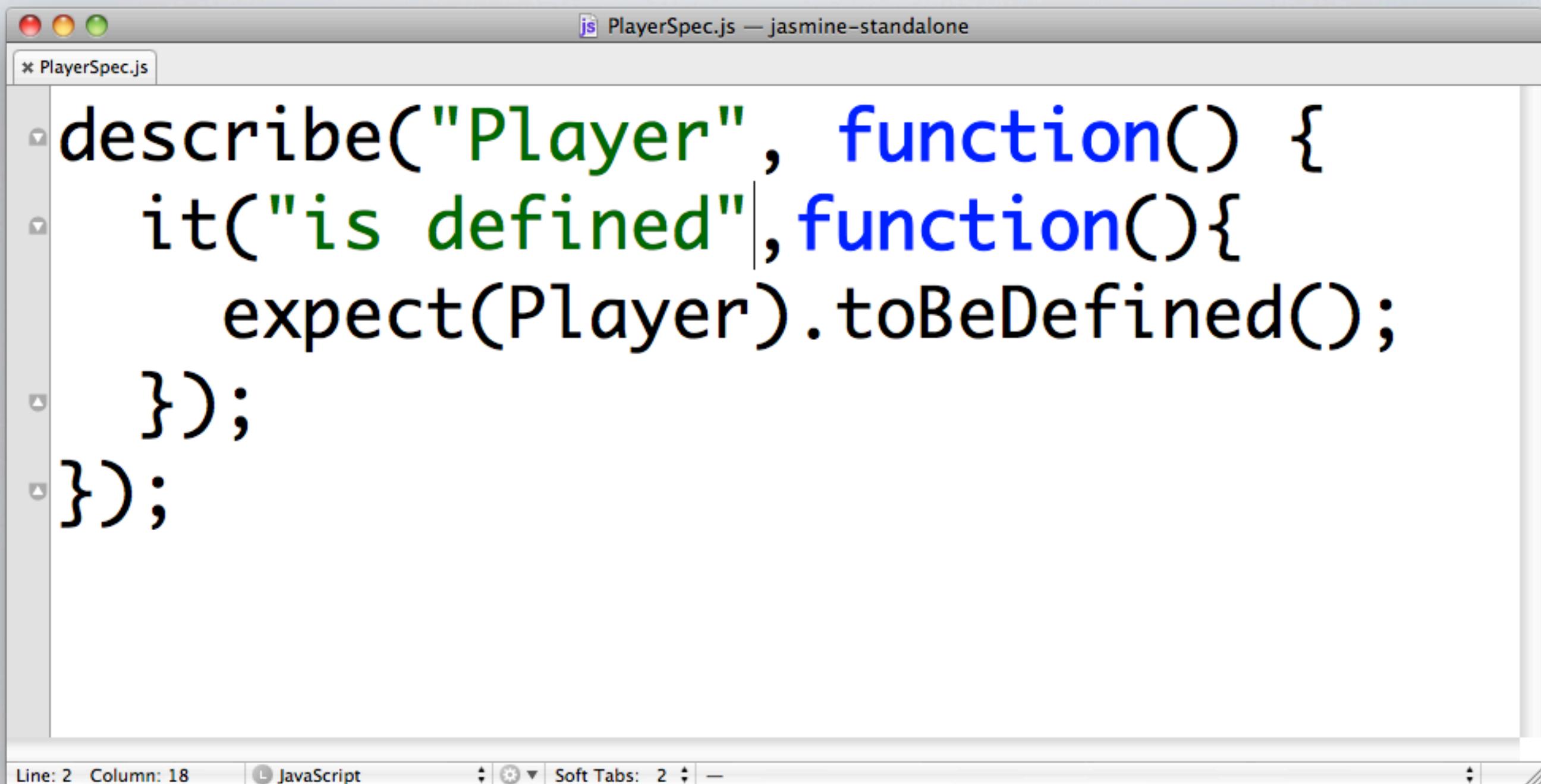
The code editor interface includes a toolbar at the top with icons for file operations, and a status bar at the bottom displaying "Line: 2 Column: 20", "JavaScript", "Soft Tabs: 2", and other settings.

## Pending examples

21

A pending example specifies something we know we want our code to do, but that we may not be implementing quite yet. If you have an existing expectation that you need to make pending, you can prepend it with an 'x', a la:

```
xit("description", function(){
  expect(code).toBePerfect();
});
```



The screenshot shows a code editor window titled "PlayerSpec.js — jasmine-standalone". The file contains the following JavaScript code:

```
describe("Player", function() {
  it("is defined", function(){
    expect(Player).toBeDefined();
  });
});
```

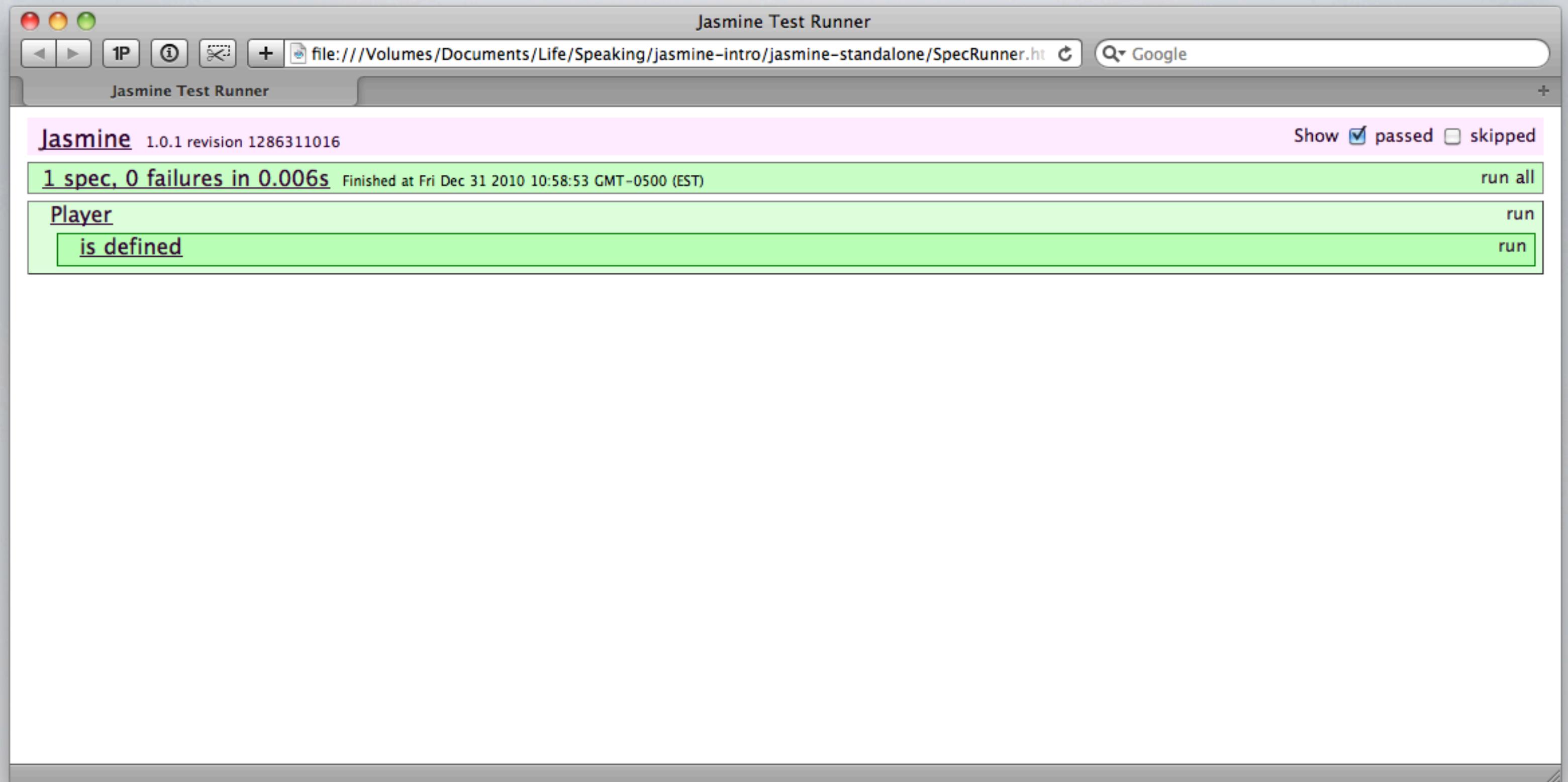
The code editor interface includes a toolbar at the top with icons for file operations, and a status bar at the bottom displaying "Line: 2 Column: 18", "JavaScript", "Soft Tabs: 2", and other settings.

## Great expectations

22

You can find all of Jasmine's default matchers here: <http://pivotal.github.com/jasmine/matchers.html>

If you're using jQuery, you can there's a great collection of handy jQuery-specific matchers too: <https://github.com/velesin/jasmine-jquery>



# Go green

23

Simply refresh the page to re-run your specs.

A fun trick is to add a meta element to your spec runner's head to refresh the page every given interval to achieve a “poor man's CI”: <meta http-equiv="refresh" content="3">

```
describe("Player", function() {
  var player;
});
```

## Nesting behavior for DRY, readable specs

24

You can nest example groups to specify cascading behavior. This can be really powerful at DRYing up your tests while retaining readability. I tend to introduce a new nested example group for each subsequent state change of my SUT that I need to specify. So that each nested example group can inherit the state of the parent, I often find myself performing arrange & act in a beforeEach() function and keep each it() focused on a single expectation regarding the behavior being specified.

The screenshot shows a web browser window with the title "PlayerSpec.js — jasmine-standalone". The tab bar at the top has two tabs: "SpecRunner.html" and "PlayerSpec.js". The main content area displays a block of JavaScript code. The code is a Jasmine specification for a "Player" object. It starts with a "describe" block for "Player", followed by a "beforeEach" block that initializes a new "Player" instance and calls its "play" method. This "beforeEach" block is nested within the "describe" block. The code ends with a closing brace for the "describe" block. The browser's status bar at the bottom shows "Line: 3 Column: 1" and "JavaScript".

```
describe("Player", function() {
  var player;
  beforeEach(function(){
    player = new Player();
    player.play();
  });

});
```

Nesting behavior for DRY, readable specs

```
describe("Player", function() {
  var player;
  beforeEach(function(){
    player = new Player();
    player.play();
  });
  it('is playing',function(){
    expect(player.isPlaying).toBeTruthy();
  })
});
```

Nesting behavior for DRY, readable specs

```
PlayerSpec.js — jasmine-standalone
* SpecRunner.html * PlayerSpec.js

describe("Player", function() {
  var player;
  beforeEach(function(){
    player = new Player();
    player.play();
  });
  it('is playing',function(){
    expect(player.isPlaying).toBeTruthy();
  })
  describe('pausing',function(){
    beforeEach(function(){ player.pause(); });
    it('is not playing',function(){
      expect(player.isPlaying).not.toBe(true);
    })
  });
});
```

Nesting behavior for DRY, readable specs

Jasmine Test Runner

file:///Volumes/Documents/Life/Speaking/jasmine-intro/jasmine-standalone/SpecRunner.html Google

velesin/ja... gist: 7072... https://gi... velesin/ja... velesin/ja... about:blank Jasmine T... searls/jas... v.gd - UR... +

**Jasmine** 1.0.1 revision 1286311016 Show  passed  skipped

**2 specs, 0 failures in 0.016s** Finished at Fri Jan 07 2011 12:24:35 GMT-0500 (EST) run all

**Player** run

pausing run

is not playing run

is playing run

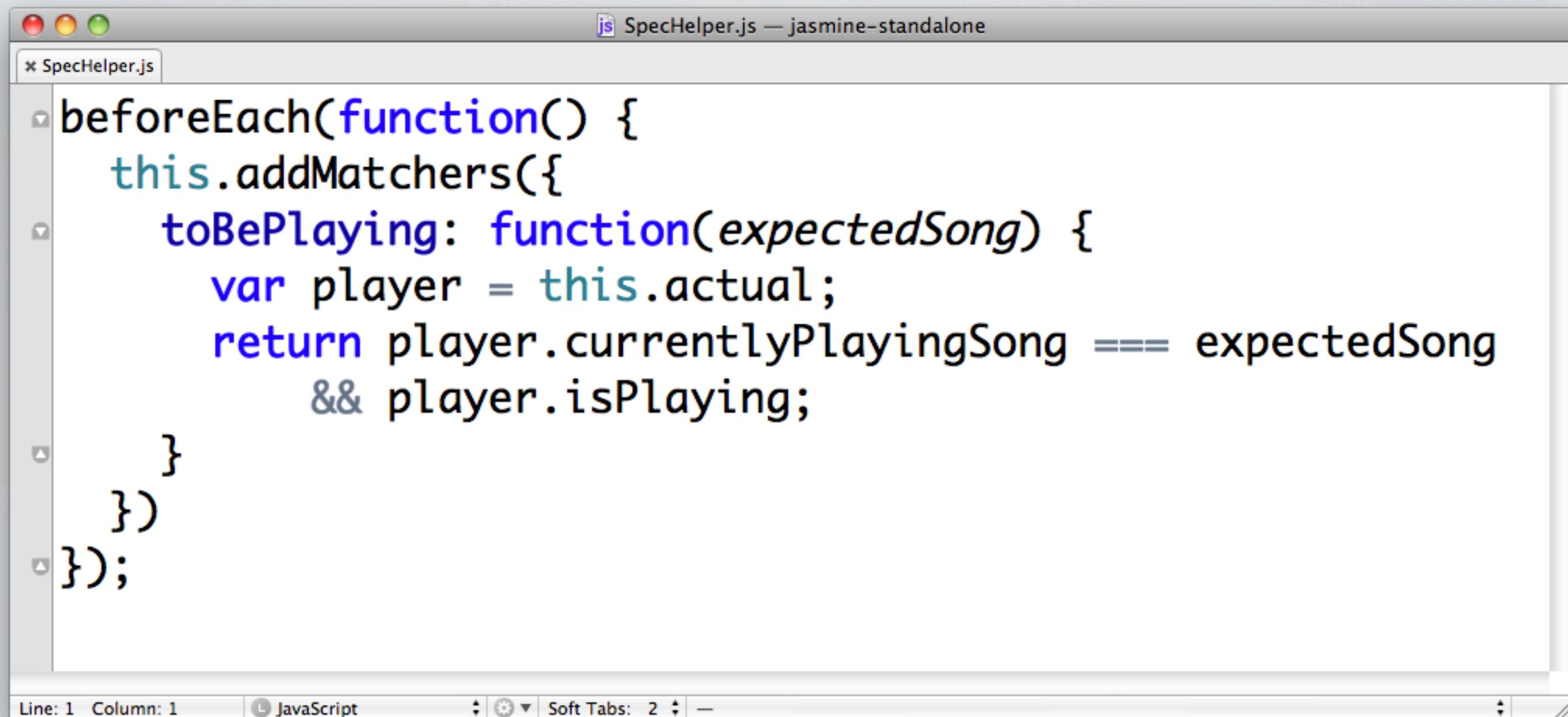
A screenshot of a web browser window titled "Jasmine Test Runner". The address bar shows the local file path. The main content area displays the test results for "Jasmine 1.0.1 revision 1286311016". It shows "2 specs, 0 failures in 0.016s" completed at "Fri Jan 07 2011 12:24:35 GMT-0500 (EST)". Below this, there's a section for the "Player" spec, which has three pending tests: "pausing", "is not playing", and "is playing", each with a "run" button next to it.

Go green

# 3. Custom Matchers

```
expect(player.isPlaying).not.toBe(true);
```

Looking back on it, that matcher is a bit tough to read. Custom matchers can be great for refactoring expectations that are too complex, will be frequently duplicated, or could do a better job of revealing their intent.



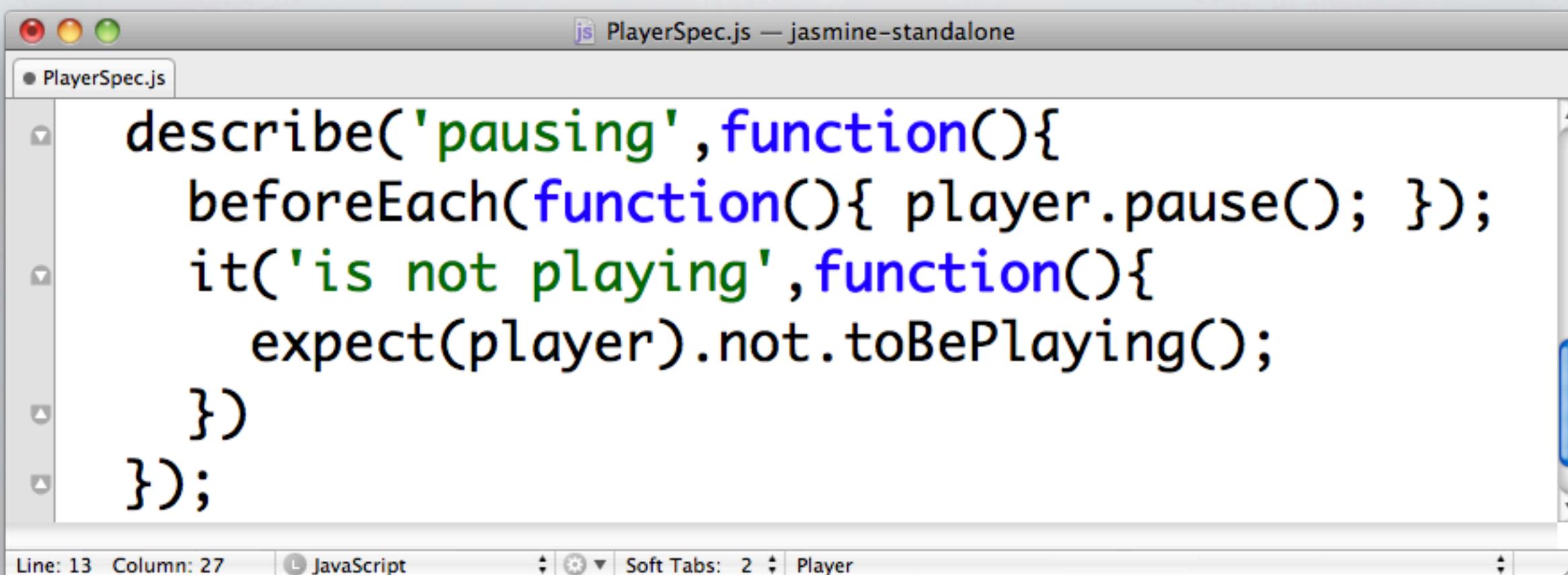
```
SpecHelper.js — jasmine-standalone
SpecHelper.js

beforeEach(function() {
  this.addMatchers({
    toBePlaying: function(expectedSong) {
      var player = this.actual;
      return player.currentlyPlayingSong === expectedSong
        && player.isPlaying;
    }
  });
});
```

Line: 1 Column: 1    JavaScript    Soft Tabs: 2

## Adding a custom matcher

It's as simple as adding an object literal of named matchers as above. You can even customize the message that's printed on expectation failures by providing a `this.message` function inside the matcher that returns a string.



The screenshot shows a code editor window titled "PlayerSpec.js — jasmine-standalone". The file content is a JavaScript test using the Jasmine framework. It defines a describe block for 'pausing' and a beforeEach block that calls player.pause(). Inside the describe block, there is an it block for 'is not playing' which uses the expect(player).not.toBePlaying() matcher.

```
describe('pausing', function(){
  beforeEach(function(){ player.pause(); });
  it('is not playing', function(){
    expect(player).not.toBePlaying();
  })
});
```

## Using a custom matcher

And any expectation can be inverted with a `.`not` prefix, without any custom work on the matcher's part

```
expect(player.isPlaying).not.toBe(true);
```

```
expect(player).not.toBePlaying();
```

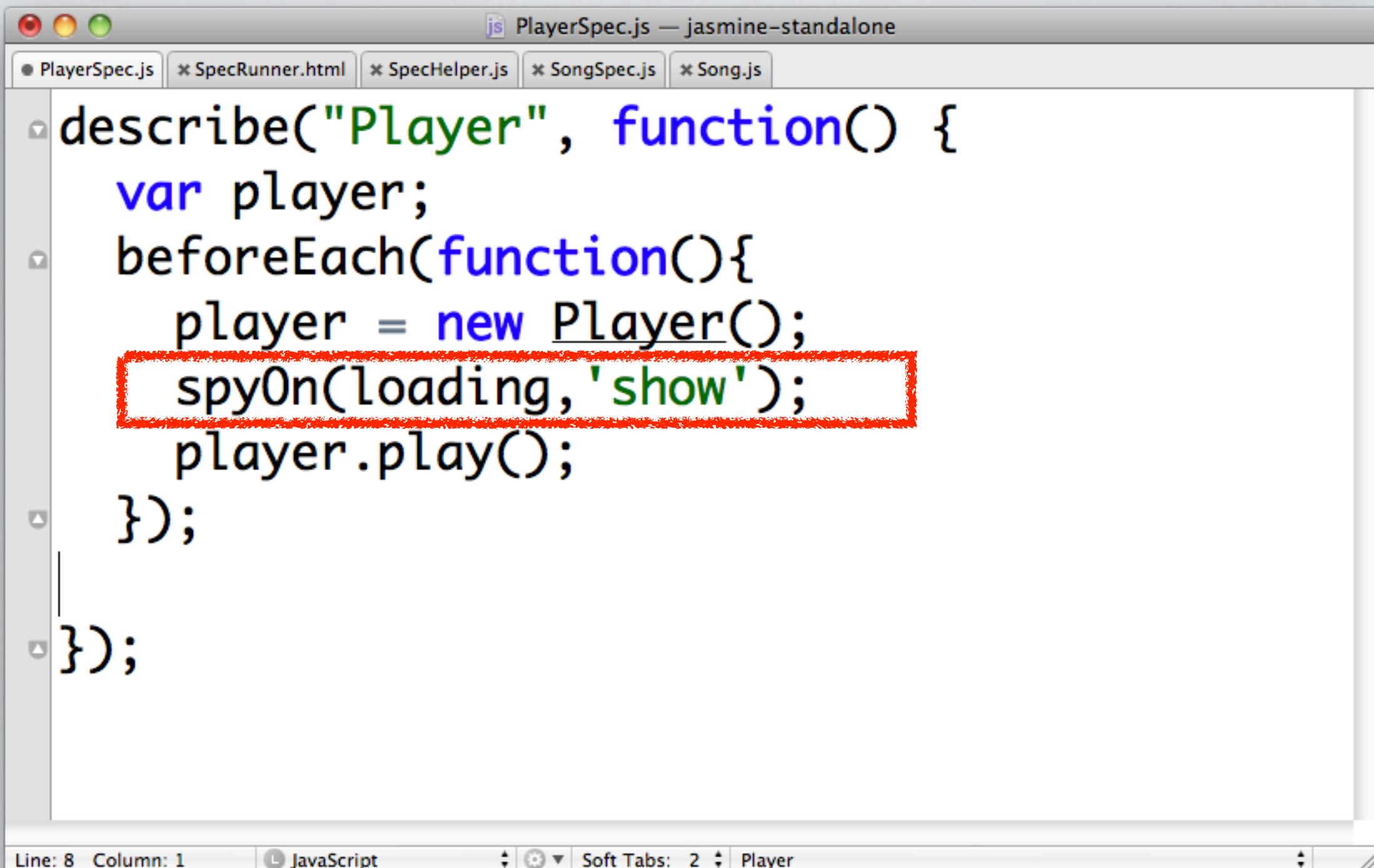
# 4. Spies

# Specify a behavior in isolation

36

Let's say that when the jukebox starts playing, a loading indicator needs to be shown.

But because the loading indicator feature is covered by specs elsewhere and animating the real thing would slow down our test execution dramatically, we want to specify only that we invoke it (isolating ourselves from what the internal logic of showing and hiding the loading indicator).

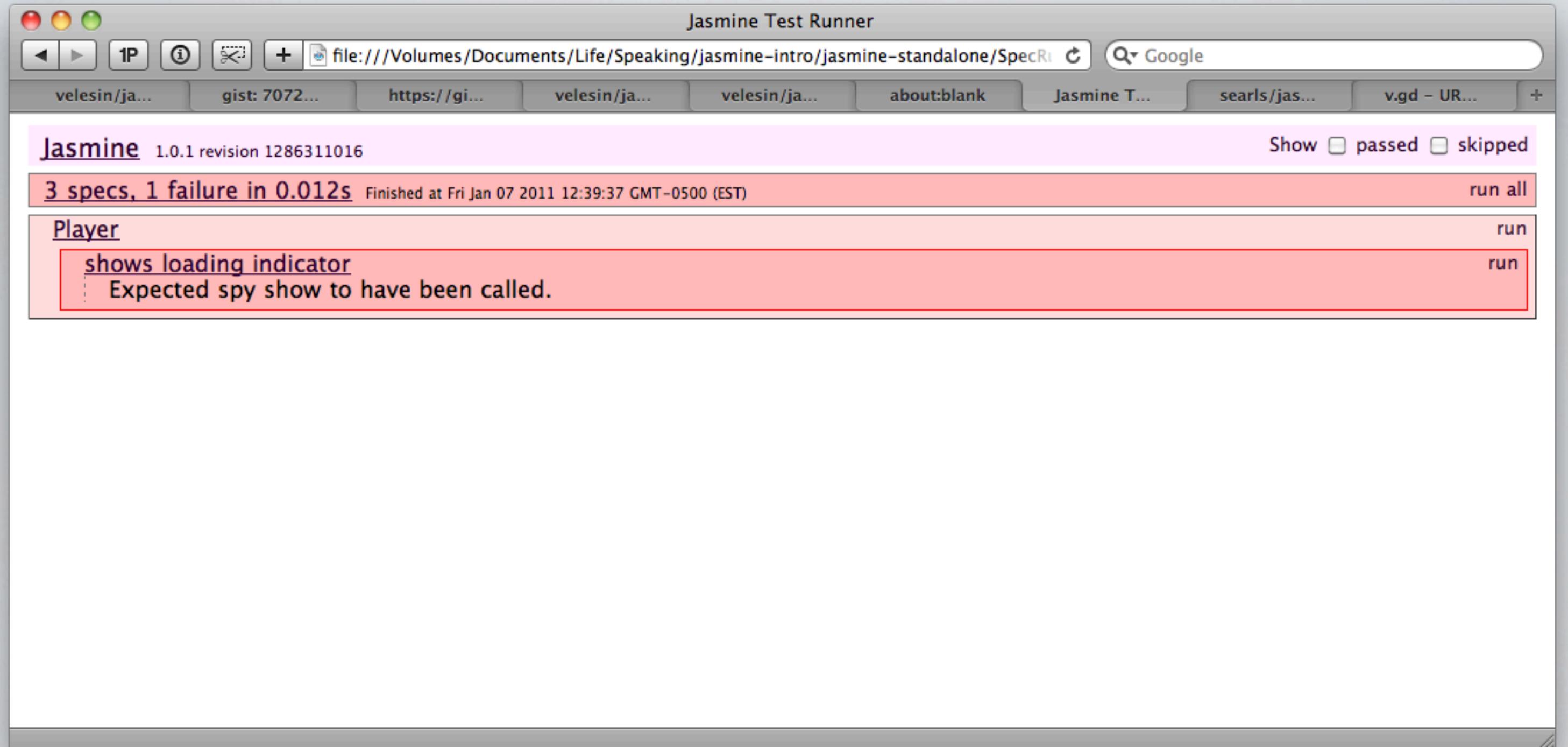


```
js PlayerSpec.js — jasmine-standalone
● PlayerSpec.js ✘ SpecRunner.html ✘ SpecHelper.js ✘ SongSpec.js ✘ Song.js
describe("Player", function() {
  var player;
  beforeEach(function(){
    player = new Player();
    spyOn(loading, 'show');
    player.play();
  });
});
```

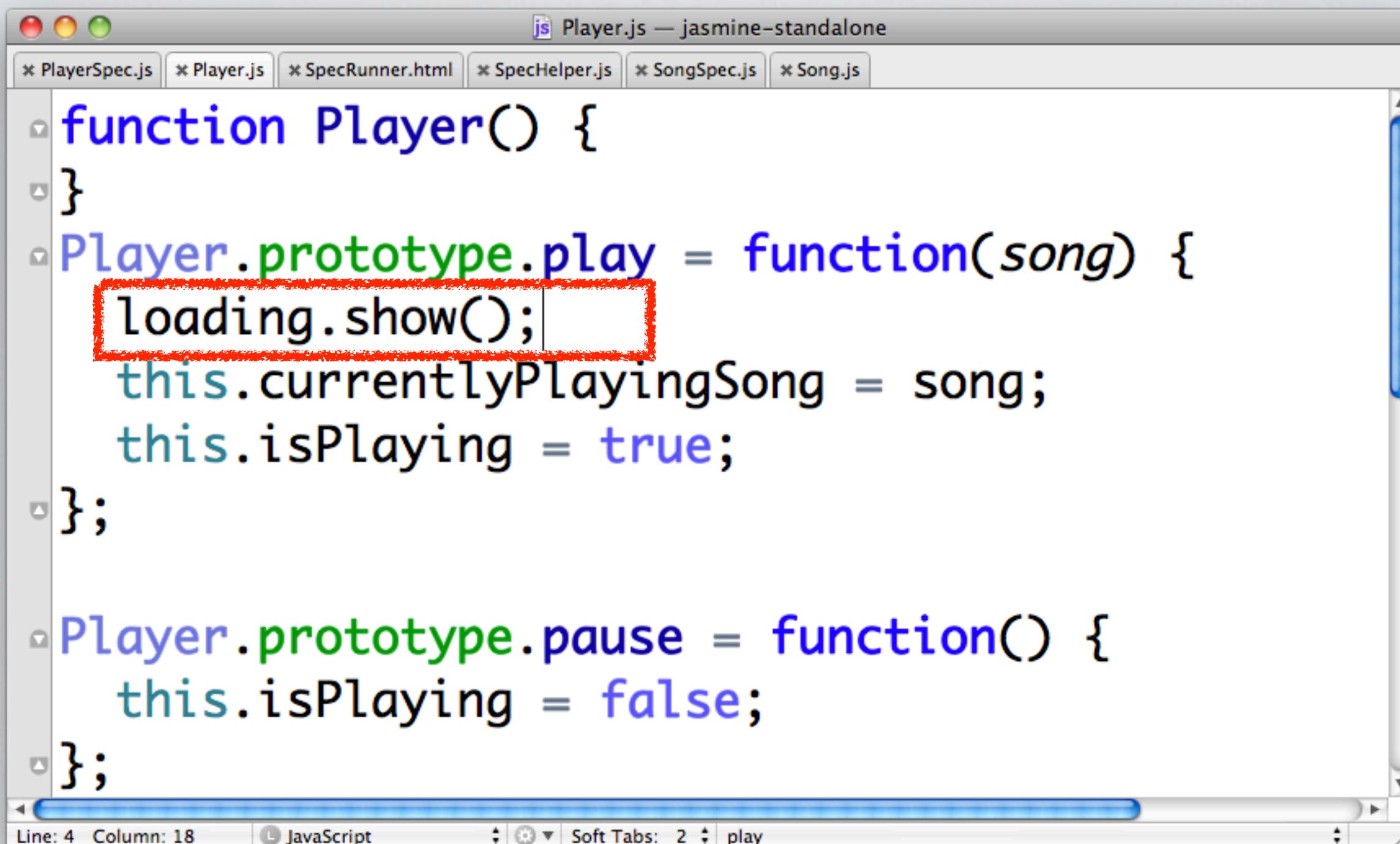
Tell Jasmine to spy on loading.show()

```
js PlayerSpec.js — jasmine-standalone
* PlayerSpec.js * Player.js * SpecRunner.html * SpecHelper.js * SongSpec.js * Song.js
describe("Player", function() {
  var player;
  beforeEach(function(){
    player = new Player();
    spyOn(loading, 'show');
    player.play();
  });
  it('shows loading indicator', function(){
    expect(loading.show).toHaveBeenCalled();
  });
});
```

Expect the invocation



Go red



```
PlayerSpec.js  * Player.js  * SpecRunner.html  * SpecHelper.js  * SongSpec.js  * Song.js
js Player.js — jasmine-standalone
function Player() {
}
Player.prototype.play = function(song) {
    loading.show();
    this.currentlyPlayingSong = song;
    this.isPlaying = true;
};

Player.prototype.pause = function() {
    this.isPlaying = false;
};
```

Line: 4 Column: 18    JavaScript    Soft Tabs: 2 play

# Implement

Jasmine Test Runner

file:///Volumes/Documents/Life/Speaking/jasmine-intro/jasmine-standalone/SpecRunner.html

Show  passed  skipped

3 specs, 0 failures in 0.011s Finished at Fri Jan 07 2011 12:41:45 GMT-0500 (EST) run all

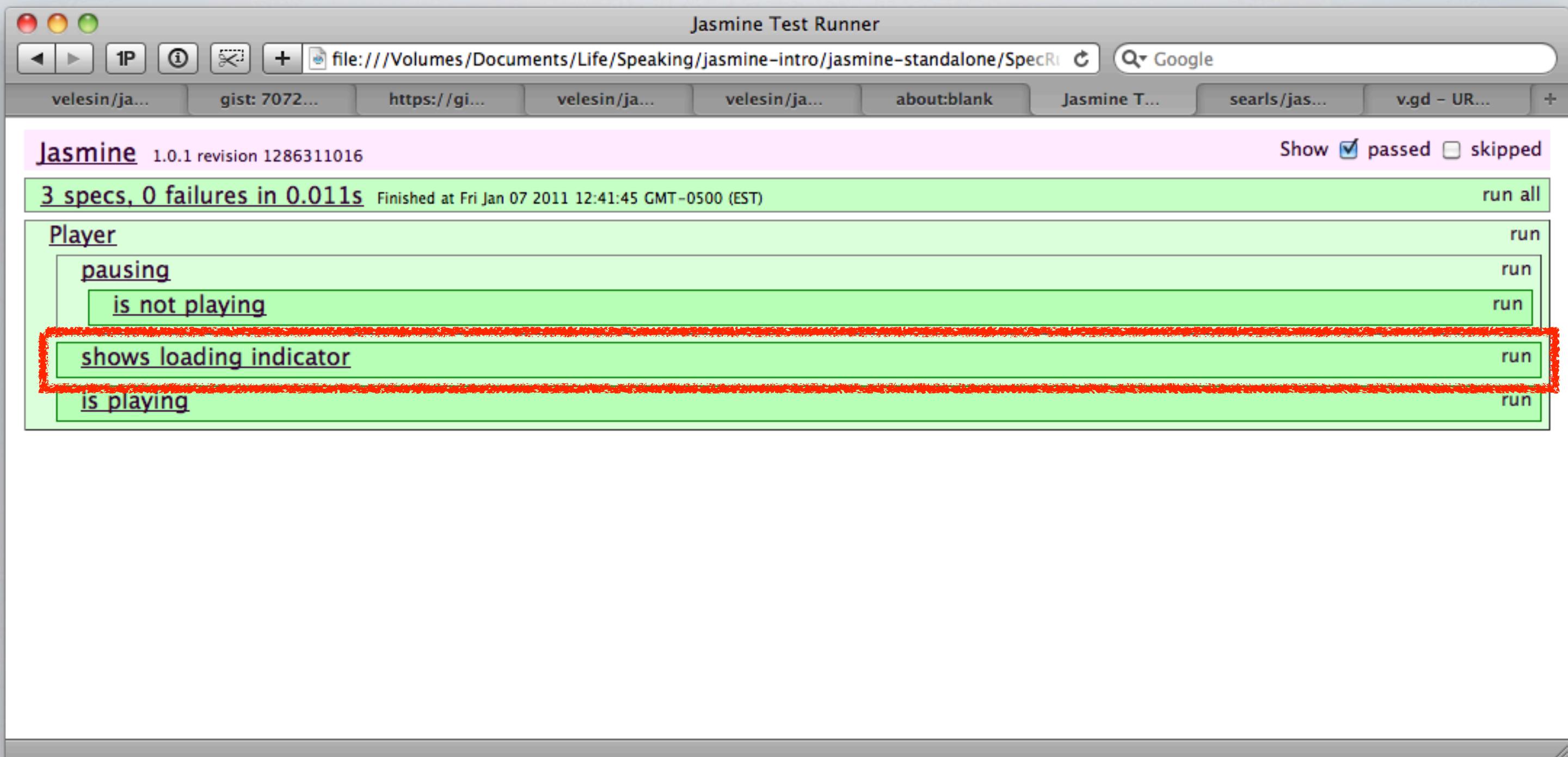
Player

pausing run

is not playing run

shows loading indicator run

is playing run



Go green!

# 5. HTML Fixtures

# “How do my specs see my HTML?”

43

This is the first question I usually hear when I people are first introduced to JavaScript testing. Most JavaScript written today is one-time-use



44

First, make an attempt to resist the temptation to dunk your delicious JavaScript in the milky DOM!

Specifying your JavaScript in a way that's independent from the exact markup it's used with has numerous benefits. Right out the gate, your JavaScript has two customers (your spec and your page), meaning that it's immediately reusable. With practice, it will probably be more modular and better abstracted, too!

# However

But, in reality, the majority of people new to JavaScript use it only for the DOM interactions that can't be performed on the server-side. Especially when you're starting out, having a way to easily and safely inject HTML fixtures into the DOM is really important.

# A lightweight script for injecting HTML

<http://v.gd/fixture>



Link expands to:  
<https://github.com/searls/jasmine-fixture>

```
describe('Song', function(){
  var song,$currentSongSpan;
  beforeEach(function(){
    song = new Song();
    song.title = 'Some Song Title';
    $.jasmine.inject('<span id="currentSong"></span>');
    $currentSongSpan = $('#currentSong');
  });
});
```

## Injecting HTML

47

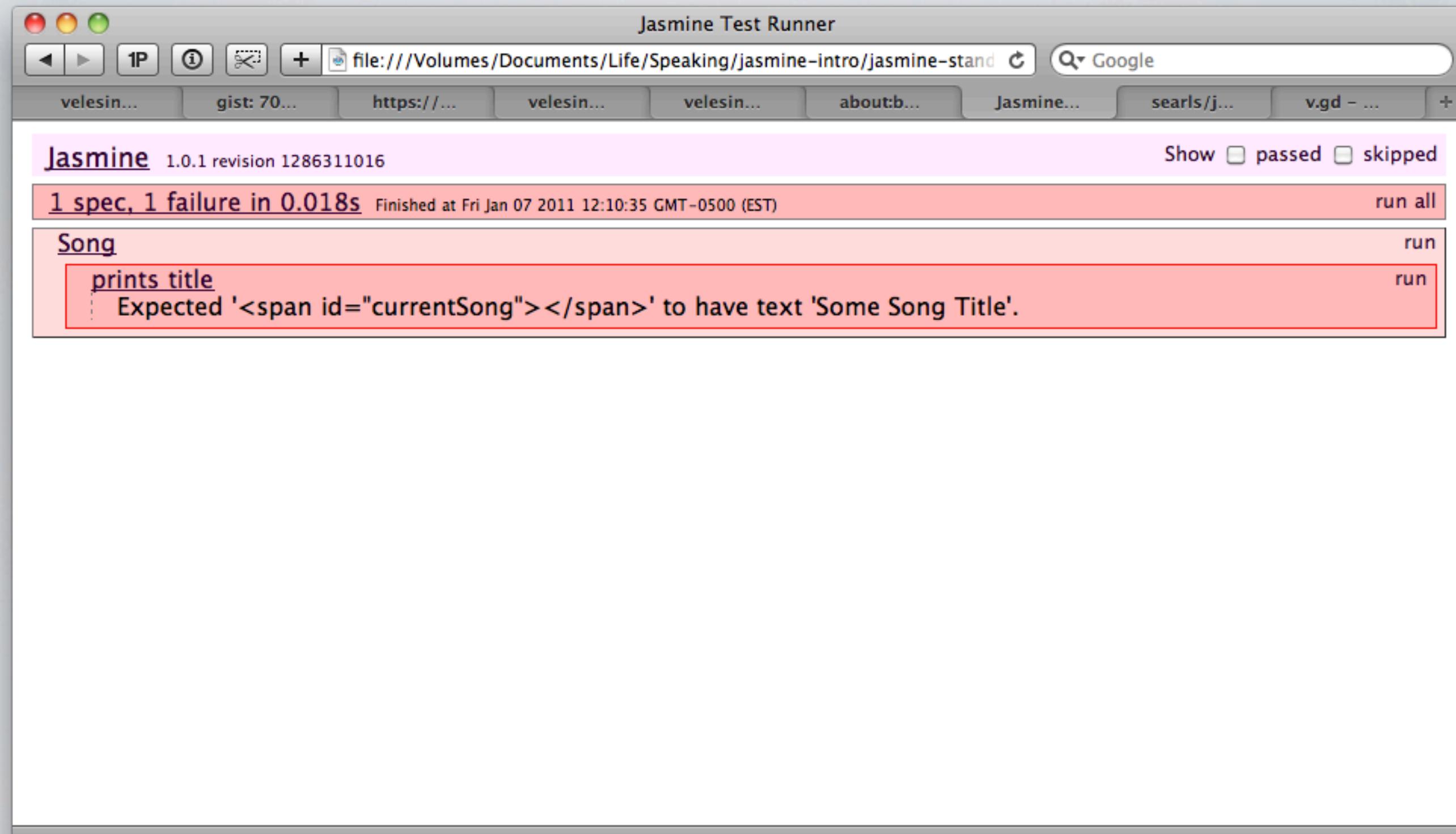
We can simply inject some HTML into the DOM to facilitate our test.

```
describe('Song', function(){
  var song,$currentSongSpan;
  beforeEach(function(){
    song = new Song();
    song.title = 'Some Song Title';
    $.jasmine.inject('<span id="currentSong"></span>');
    $currentSongSpan = $('#currentSong');
  });
  it('prints title', function(){
    song.printTitle();
    expect($currentSongSpan).toHaveText(song.title);
  })
});
```

## Injecting HTML

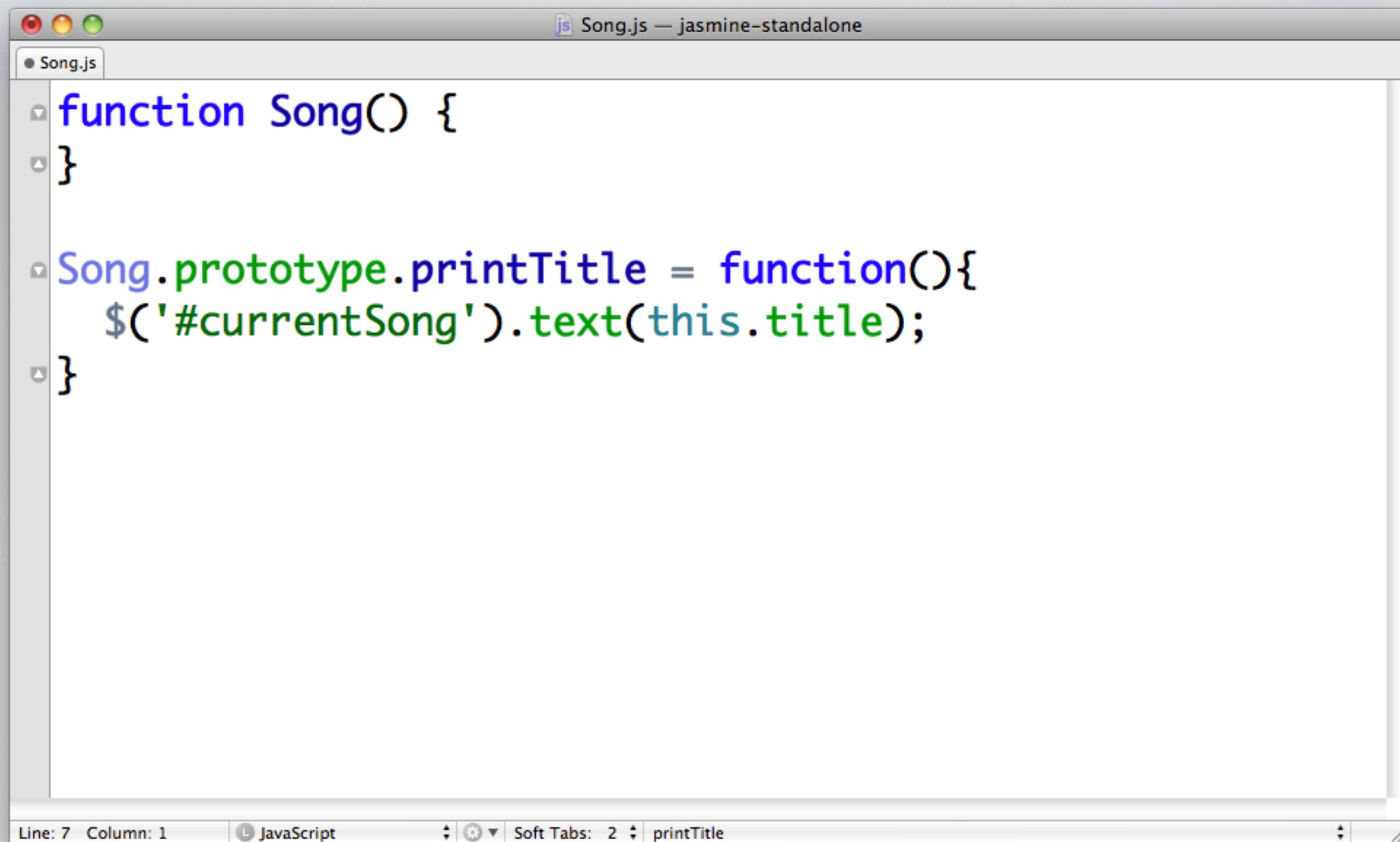
48

This way we can confidently specify the behavior of code interacting with the DOM, while minimizing the coupling to specific markup.



# Go red

Now we have a nice clear failure.



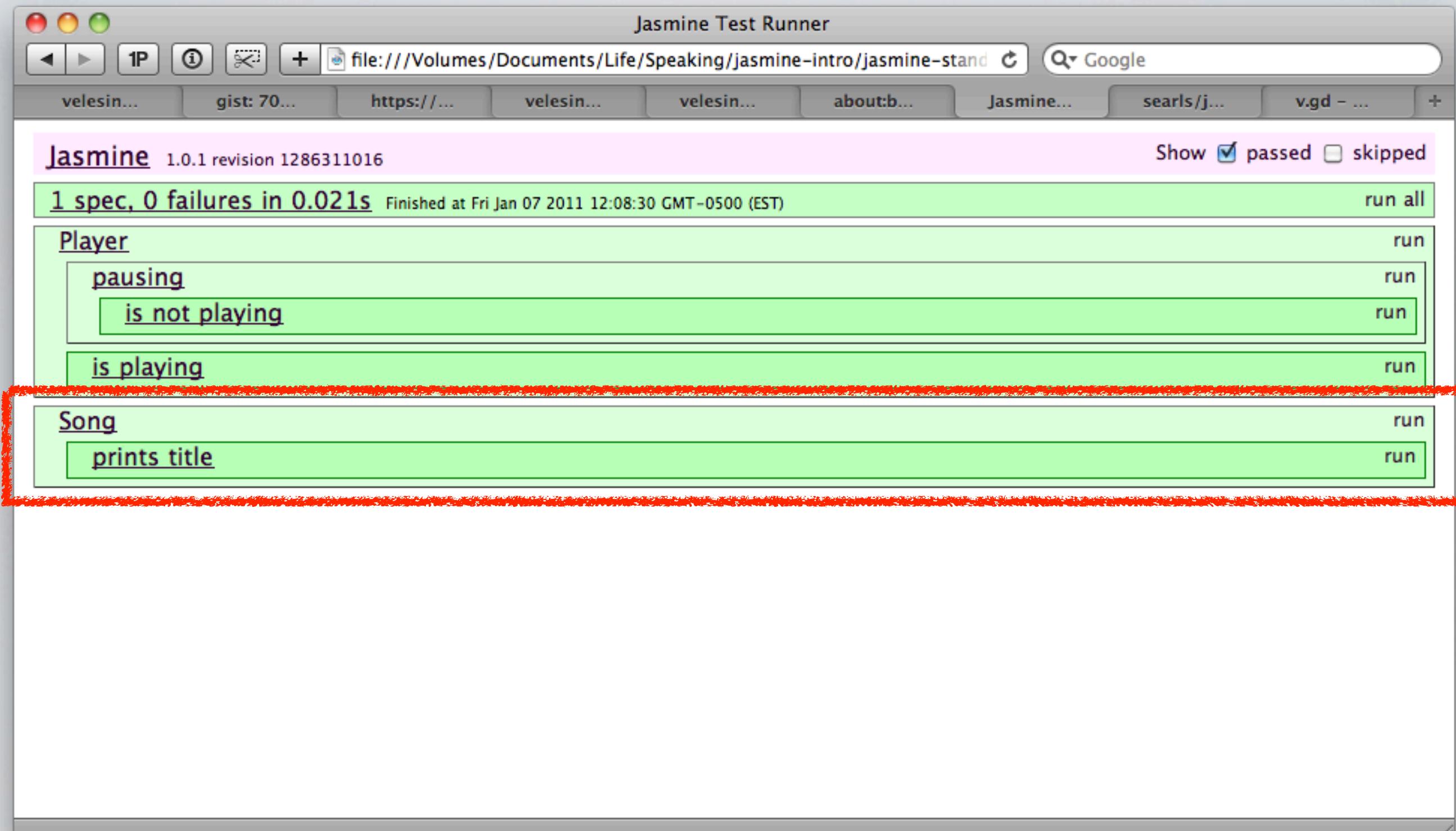
A screenshot of a code editor window titled "Song.js — jasmine-standalone". The file "Song.js" is open. The code defines a function Song() and its prototype printTitle(). The printTitle() function uses jQuery to set the text of an element with id "currentSong" to the title of the song.

```
function Song() {  
}  
  
Song.prototype.printTitle = function(){  
    $('#currentSong').text(this.title);  
}
```

## Injecting HTML

50

Now our implementation is free to interact with the specific DOM element our test provides



# Go green!

And our spec goes green!

# 6. Continuous Integration / Project Comprehension

pivotal/jasmine-gem - GitHub

git https://github.com/pivotal/jasmine-gem

RSS Google

pivotal/jasmine-gem - GitHub

**github**  
SOCIAL CODING

searls 14 | Dashboard | Inbox 6 | Account Settings | Log Out

Explore GitHub Gist Blog Help Search...

Watch Fork 110 40

⌚ pivotal / **jasmine-gem**

Source Commits Network Pull Requests (5) Graphs Branch: master

Switch Branches (3) ▾ Switch Tags (28) ▾ Branch List

Jasmine ruby gem — [Read more](#) Downloads

HTTP Git Read-Only <https://github.com/pivotal/jasmine-gem.git>  This URL has **Read-Only** access

Adding gem-release gem as a dev dependency

 Davis W. Frank (author)

December 07, 2010

commit 143177702382a5516d5  
tree 2631ff0358652e0f3fd6  
parent 5e14fc53ba5f10553d37

**jasmine-gem /**

name	age	message	history
------	-----	---------	---------

# jasmine-gem for Ruby/Rails under Selenium

pivotalexperimental/jazz\_money - GitHub

git https://github.com/pivotalexperimental/jazz\_money

searls 14 | Dashboard | Inbox 6 | Account Settings | Log Out

Explore GitHub Gist Blog Help Search...

Watch Fork 22 12

Source Commits Network Pull Requests (0) Issues (1) Graphs Branch: master

Switch Branches (1) Switch Tags (0) Branch List

Run your Jasmine specs without launching a browser — [Read more](#)

Downloads

HTTP Git Read-Only https://github.com/pivotalexperimental/jazz\_mon  This URL has **Read-Only** access

README update

Mike Grafton (author) October 30, 2010

commit cfa30a9cd212446d62b6  
tree 80501fc6d0dcfb77095a  
parent ecf5b73ee40c168c7223

jazz\_money /

name	age	message	history
------	-----	---------	---------

# jazz\_money for headless Ruby

searls/jasmine-maven-plugin - GitHub

git https://github.com/searls/jasmine-maven-plugin

searls/jasmine-maven-plugin - ...

searls 14 | Dashboard | Inbox 6 | Account Settings | Log Out

Explore GitHub Gist Blog Help

Search...

searls / jasmine-maven-plugin

Admin Unwatch Fork Pull Request 23 11

Source Commits Network Pull Requests (0) Fork Queue Issues (2) Wiki (1) Graphs Branch: master

Switch Branches (2) Switch Tags (0) Branch List

Maven plugin to execute Jasmine Specs. Creates your HTML runners for you, runs headlessly, outputs JUnit XML — [Read more](#)

<http://about.me/searls>

Downloads

SSH HTTP Git Read-Only git@github.com:searls/jasmine-maven-plugin.git  This URL has **ReadWrite** access

doc

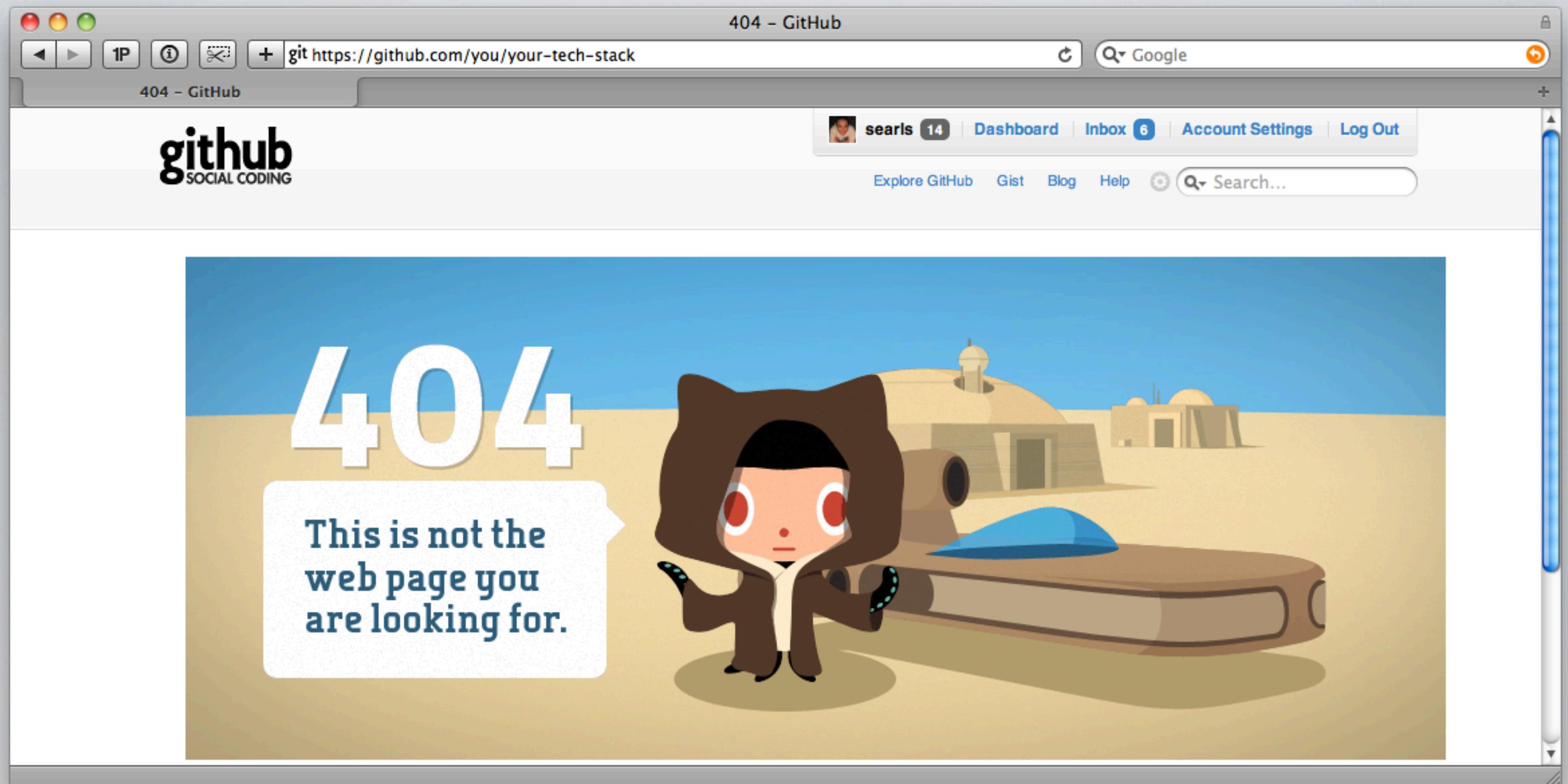
searls (author)  
December 23, 2010

commit 0e8a8ce41eedb1cf318f  
tree b8ea48d398d229737dfe  
parent 91050e19218c847d09a0

jasmine-maven-plugin /

name age message history

# jasmine-maven-plugin for Java projects



Let's build tools for your tech stack together!

# Image Credits

- ★ <http://www.sxc.hu/photo/708473>
- ★ <http://www.sxc.hu/photo/859430>
- ★ <http://www.sxc.hu/photo/1103790>
- ★ <http://www.sxc.hu/photo/1309077>
- ★ <http://www.flickr.com/photos/paperdollimages/512768518/>  
(via [\\*!·sindorella·!\\*](#))