

Jasmine

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

pillar

@searls

<http://github.com/searls/jasmine-intro>

You can find this presentation & code at:

http://v.gd/jasmine_intro



Who?



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
Current Build Status: [Jasmine at Pivotal Labs CI](#)
Report bugs at [GitHub](#)
Project Backlog: [Jasmine on Pivotal Tracker](#)

Recent Activity:

-  **quigebo** writing JS tests for the first time w/ Jasmine - <https://github.com/pivotal/jasmine/tree/master/spec/javascript> - and loving it. Very RSpec-y
3 days ago · reply
-  **krismolendyke** @CoffeeScript @jasminebdd @jquery Whoa

When?



11

As soon as you're ready to jump in

Why?



13

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

More on Why:

<http://v.gd/javascript>



14

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

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](#)

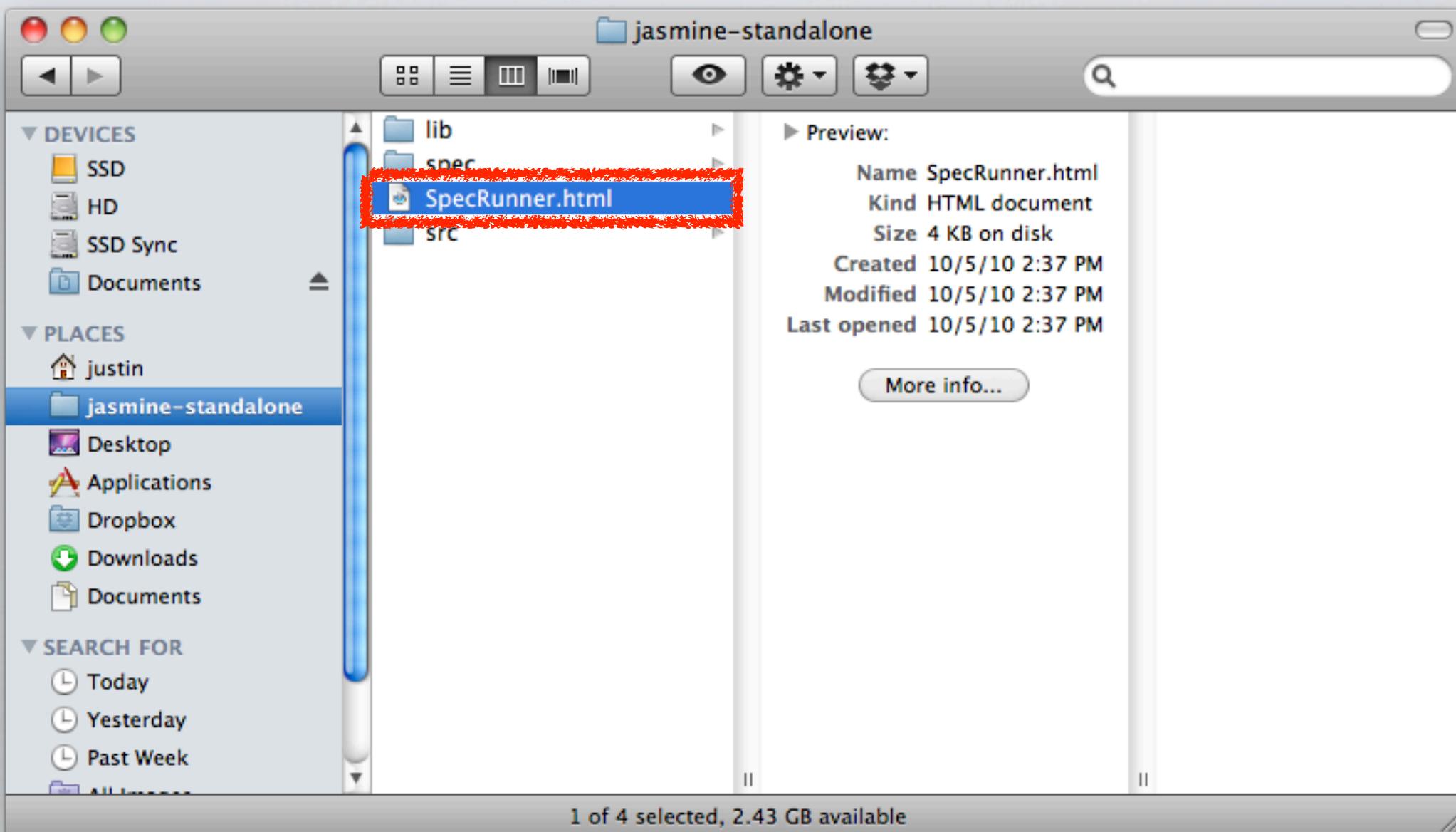
Downloads

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
jasmine-standalone-1.0.1.zip	1.0.1	20k	2010/10/05 13:37:01 PDT	b2b3d00a7cb8d5cccd65a3356bb5ae15775328119
jasmine-standalone-1.0.0.zip	1.0.0	20k	2010/09/14 12:54:38 PDT	1866f654a3ad0ab9109393ce31d6613c77916607
jasmine-standalone-1.0.0.rc1.zip	1.0.0.rc1	19k	2010/08/26 13:09:41 PDT	20e7da22bc7ce3433331a5ad44eb199f4ff34065
jasmine-standalone-0.11.1.zip	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

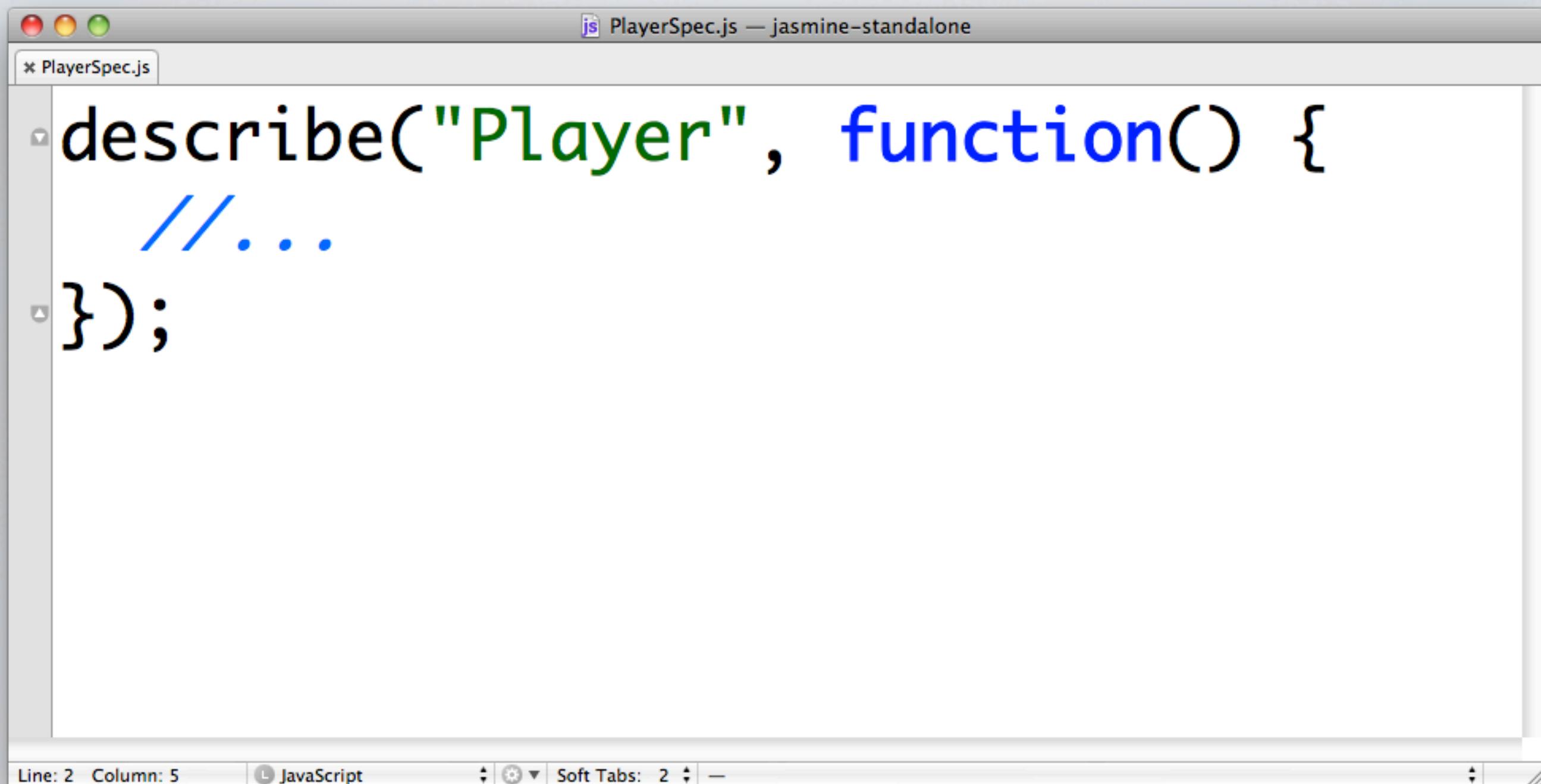
run

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:

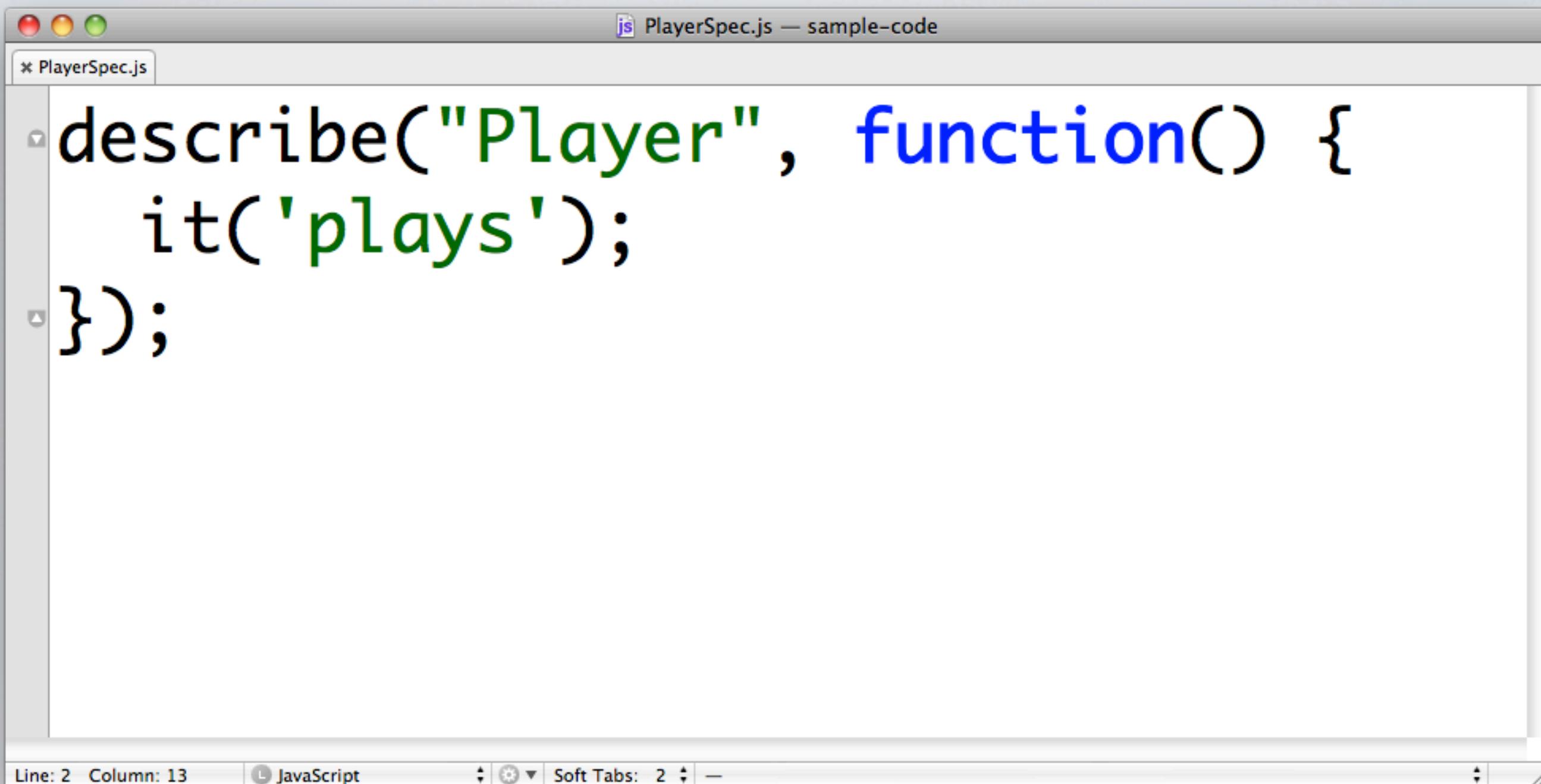
```
describe("Player", function() {  
    //...  
});
```

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!

21

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 — sample-code". The file "PlayerSpec.js" is open, showing the following code:

```
describe("Player", function() {
  it('plays');
});
```

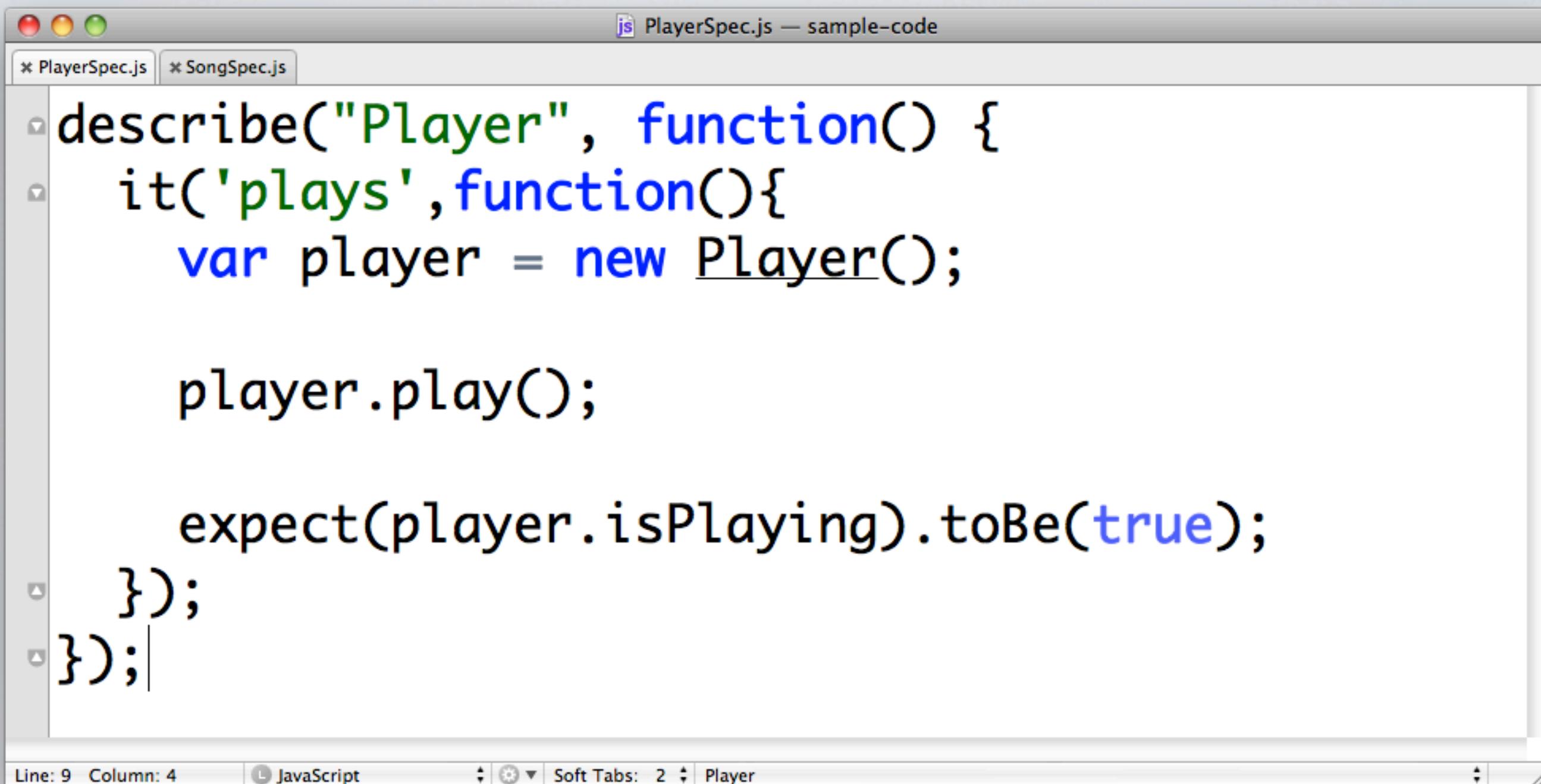
The code editor interface includes a toolbar at the top with red, yellow, and green buttons, and a status bar at the bottom displaying "Line: 2 Column: 13", "JavaScript", "Soft Tabs: 2", and other icons.

Pending examples

22

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();
});
```



```
PlayerSpec.js — sample-code
PlayerSpec.js SongSpec.js
describe("Player", function() {
  it('plays', function(){
    var player = new Player();
    player.play();
    expect(player.isPlaying).toBe(true);
  });
});
```

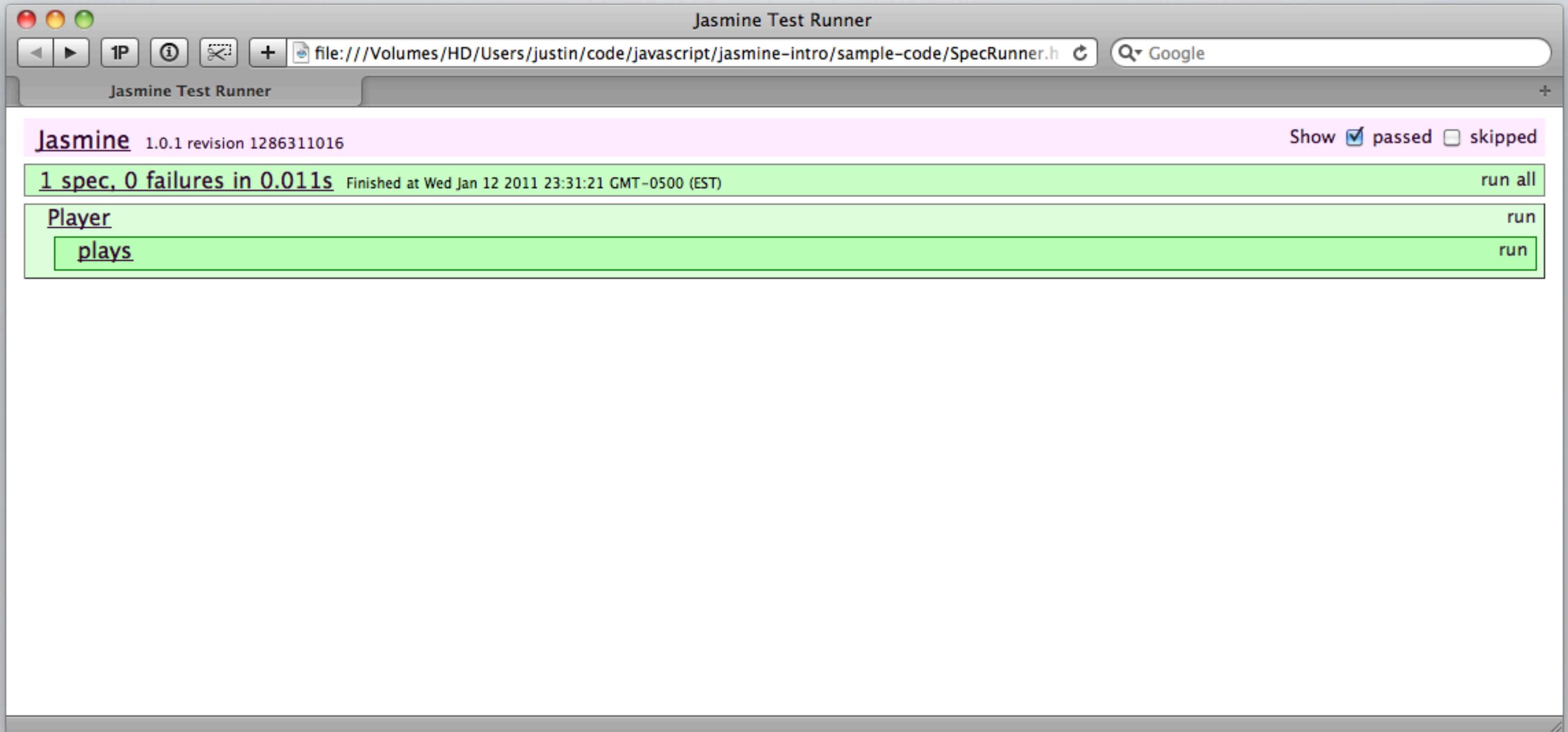
Line: 9 Column: 4 JavaScript Soft Tabs: 2 Player

Great expectations

23

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>



Green!

Simply refresh the page to re-run your specs.

A fun trick is to automatically refresh the page every given interval to implement a rudimentary autotest: <meta http-equiv="refresh" content="3">

The screenshot shows a terminal window with a title bar "PlayerSpec.js — sample-code". The window contains two tabs: "PlayerSpec.js" (selected) and "SongSpec.js". The code in "PlayerSpec.js" is as follows:

```
});  
it('pauses', function() {  
  var player = new Player();  
  player.play();  
  
  player.pause();  
  
  expect(player.isPlaying).not.toBe(true);  
})  
});
```

The terminal status bar at the bottom shows "Line: 1 Column: 32" and "JavaScript".

Our second example

Let's create a similar spec for the player's pause function

Jasmine Test Runner

Jasmine Test Runner

Jasmine 1.0.1 revision 1286311016

Show passed skipped

2 specs, 0 failures in 0.01s Finished at Wed Jan 12 2011 23:18:40 GMT-0500 (EST)

run all

Player

plays

pauses

run

run

run

Green!

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

```
describe("Player", function() {
  it('plays', function(){
    var player = new Player();

    player.play();

    expect(player.isPlaying).toBe(true);
  });

  it('pauses', function() {
    var player = new Player();
    player.play();

    player.pause();

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

duplication!

Let's create a similar spec for the player's pause function

```
js PlayerSpec.js — sample-code
PlayerSpec.js SongSpec.js

describe("Player", function() {
  it('plays', function(){
    var player = new Player();
    player.play();

    expect(player.isPlaying).toBe(true);
  });
  it('pauses', function() {
    var player = new Player(); ←
    player.play();

    player.pause();

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

duplication!

Let's create a similar spec for the player's pause function

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

```
describe("Player", function() {
  it('plays', function(){
    var player = new Player();

    player.play();

    expect(player.isPlaying).toBe(true);
  });

  it('pauses', function() {
    var player = new Player();
    player.play();

    player.pause();

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

duplication!

Let's create a similar spec for the player's pause function

PlayerSpec.js — sample-code

* PlayerSpec.js * SongSpec.js

```
describe("Player", function() {
  it('plays', function(){
    var player = new Player();
    player.play();
    expect(player.isPlaying).toBe(true);
  });
  it('pauses', function() {
    var player = new Player();
    player.play();
    player.pause();
    expect(player.isPlaying).not.toBe(true);
  })
});
```

Line: 17 Column: 4 | L JavaScript | Soft Tabs: 2 | Player

duplication!

Let's create a similar spec for the player's pause function

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

```
describe("Player", function() {
  it('plays', function(){
    var player = new Player();

    player.play();

    expect(player.isPlaying).toBe(true);
  });

  it('pauses', function() {
    var player = new Player();
    player.play();

    player.pause();

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

duplication!

Let's create a similar spec for the player's pause function



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

nesting

In Jasmine, 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 whenever I need to specify the behavior of code following a state change (in this case, playing the Player). 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.

Jasmine Test Runner

file:///Volumes/HD/Users/justin/code/javascript/jasmine-intro/sample-code/SpecRunner.htm Google

Jasmine Test Runner

Jasmine 1.0.1 revision 1286311016 Show passed skipped

2 specs, 0 failures in 0.014s Finished at Wed Jan 12 2011 23:18:49 GMT-0500 (EST) run all

Player

pausing run

pauses run

plays run

A screenshot of a web browser window titled "Jasmine Test Runner". The address bar shows the local file path "file:///Volumes/HD/Users/justin/code/javascript/jasmine-intro/sample-code/SpecRunner.htm". The main content area displays the test results. At the top left, it says "Jasmine 1.0.1 revision 1286311016". To the right is a "Show" button followed by two checkboxes: one for "passed" (which is checked) and one for "skipped". Below this, a green header bar shows "2 specs, 0 failures in 0.014s" and the date "Finished at Wed Jan 12 2011 23:18:49 GMT-0500 (EST)". On the far right of this bar is a "run all" button. The main body of the page is titled "Player". It contains three test cases: "pausing", "pauses", and "plays", each with a "run" button to its right. All three test cases are listed in green, indicating they have passed.

Still 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.

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

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

Line: 8 Column: 4 JavaScript Soft Tabs: 2 toBePlaying

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.

```
PlayerSpec.js — sample-code
PlayerSpec.js

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

Using our custom matcher

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

Jasmine Test Runner

file:///Volumes/HD/Users/justin/code/javascript/jasmine-intro/sample-code/SpecRunner.htm Google

Jasmine Test Runner

Jasmine 1.0.1 revision 1286311016 Show passed skipped

2 specs, 0 failures in 0.014s Finished at Wed Jan 12 2011 23:18:49 GMT-0500 (EST) run all

Player

pausing run

pauses run

plays run

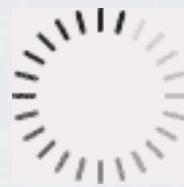
This screenshot shows the Jasmine Test Runner interface in a web browser. The title bar reads "Jasmine Test Runner". The address bar shows the local file path "file:///Volumes/HD/Users/justin/code/javascript/jasmine-intro/sample-code/SpecRunner.htm" and the search term "Google". The main content area displays the test results. At the top left, it says "Jasmine 1.0.1 revision 1286311016". To the right are buttons for "Show" (with a checked checkbox for "passed" and an unchecked checkbox for "skipped"), "run all", and a "+" sign. Below this, a green header bar shows "2 specs, 0 failures in 0.014s" and the timestamp "Finished at Wed Jan 12 2011 23:18:49 GMT-0500 (EST)". The main body lists a single suite named "Player". Under "Player", there are three test cases: "pausing", "pauses", and "plays", each with a "run" button to its right. All three test cases are listed in green, indicating they have passed.

Still green

4. Spies

Specify a behavior

Specify a loading animation



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).

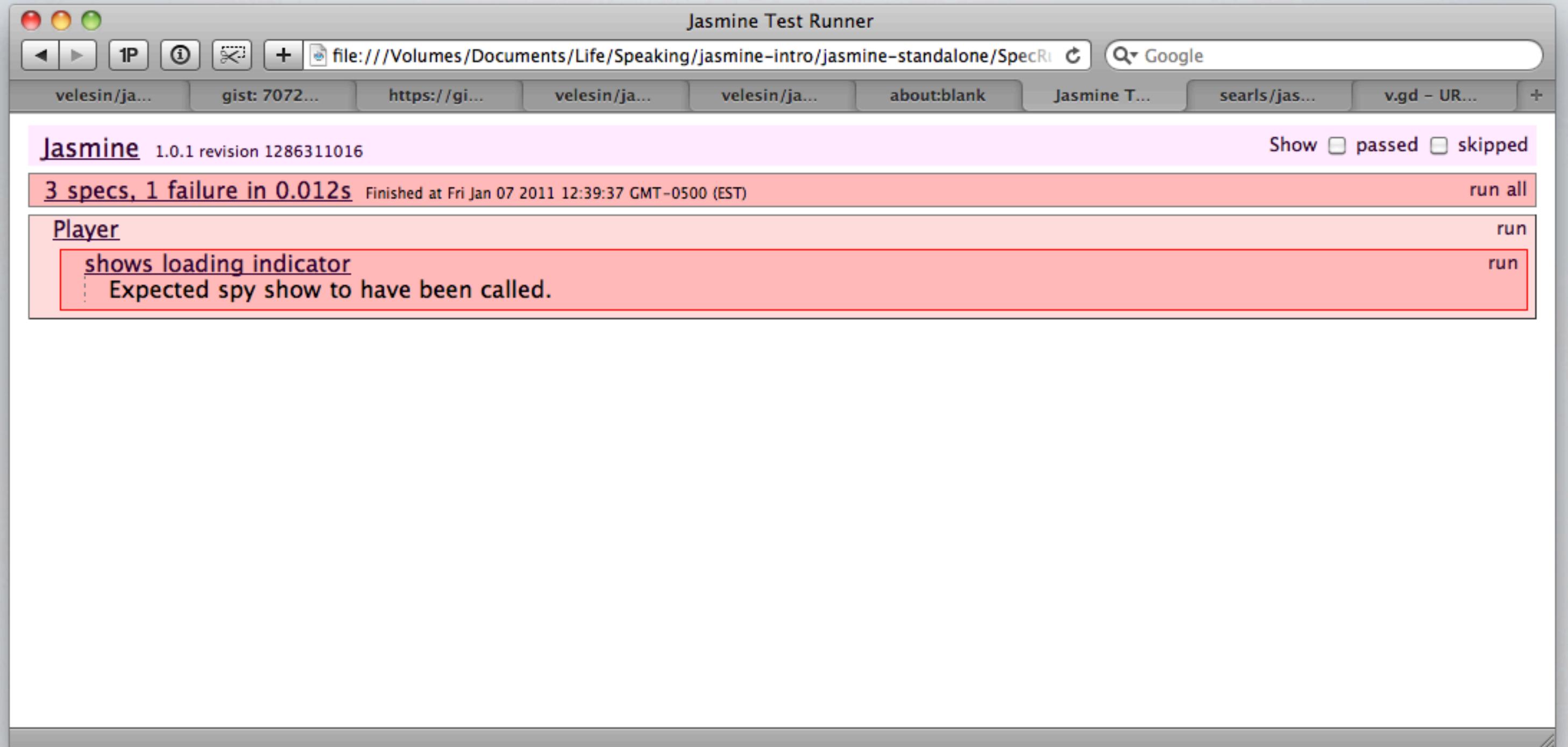
The screenshot shows a code editor window titled "PlayerSpec.js — sample-code". The file content is a JavaScript test using the Jasmine framework. It defines a describe block for "Player" and a beforeEach block. Inside the beforeEach block, a new instance of the Player class is created, and the loading.show method is spied on. The spyOn call is highlighted with a red box. The test then checks if the player is playing.

```
describe("Player", function() {
  var player;
  beforeEach(function(){
    player = new Player();
    spyOn(loading, 'show');
    player.play();
  });
  it('plays', function(){
    expect(player.isPlaying).toBe(true);
  });
});
```

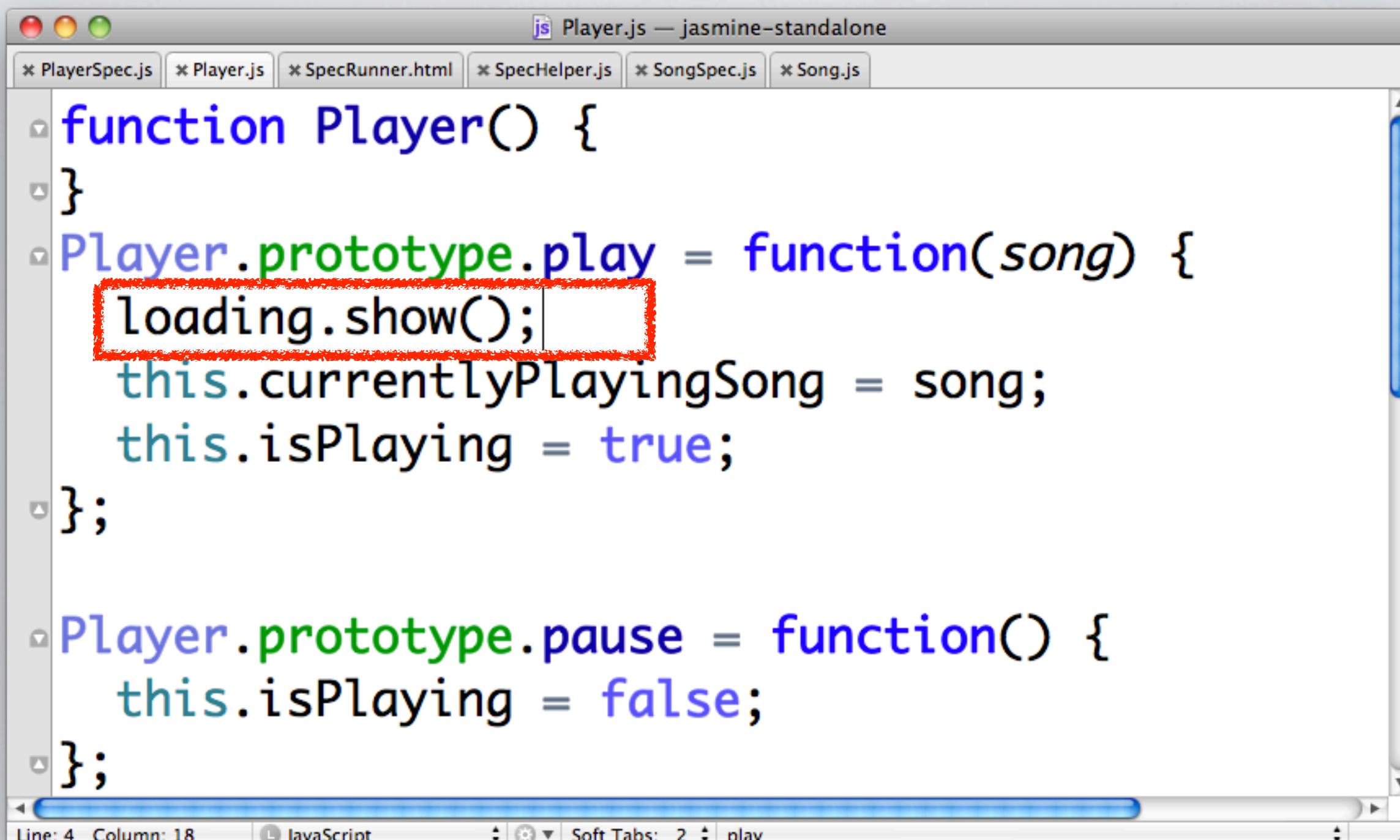
Tell Jasmine to spy on loading.show()

```
PlayerSpec.js — sample-code
* PlayerSpec.js
describe("Player", function() {
  var player;
  beforeEach(function(){
    player = new Player();
    spyOn(loading, 'show');
    player.play();
  });
  it('shows a loading indicator', function(){
    expect(loading.show).toHaveBeenCalled();
  });
})
Line: 12 Column: 41 | L JavaScript | Soft Tabs: 2 | Player
```

Expect the invocation



Go red



```
function Player() {
}
Player.prototype.play = function(song) {
    loading.show();
    this.currentlyPlayingSong = song;
    this.isPlaying = true;
};

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

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

	run
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?”

46

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



47

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>

The screenshot shows a code editor window with the title "SongSpec.js — jasmine-standalone". The file contains a Jasmine specification for a "Song" object. It includes a "beforeEach" block that creates a new "Song" instance, sets its "title" to "Some Song Title", and injects an HTML span element with the id "currentSong" into the DOM. The injected HTML is highlighted with a red box.

```
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

50

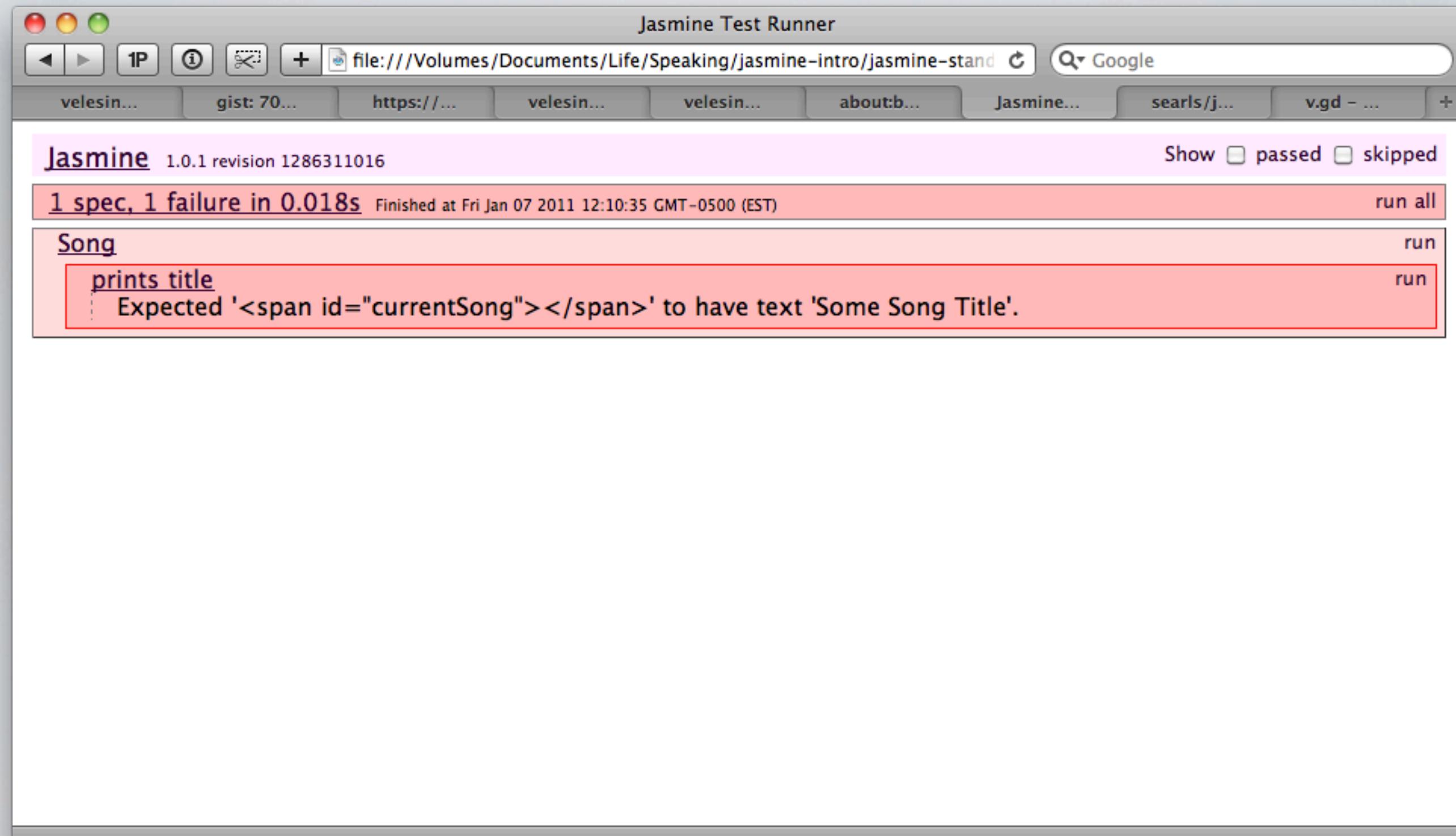
We can simply inject some HTML into the DOM to facilitate our test.

```
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

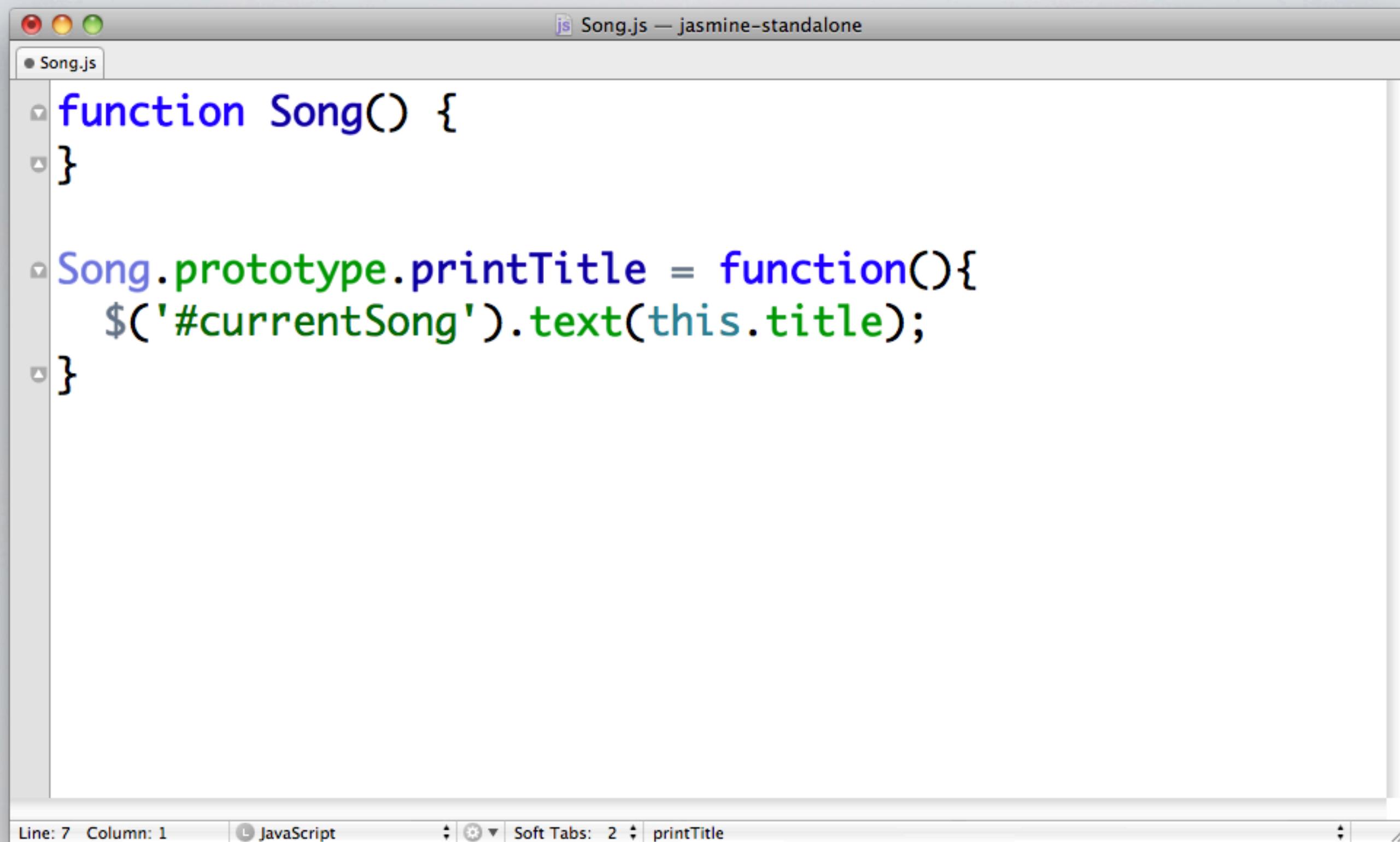
51

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

53

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

Jasmine Test Runner

file:///Volumes/Documents/Life/Speaking/jasmine-intro/jasmine-stand Google

velesin... gist: 70... https://... velesin... velesin... about:b... Jasmine... searls/j... v.gd - ... +

Jasmine 1.0.1 revision 1286311016

Show passed skipped

1 spec, 0 failures in 0.021s Finished at Fri Jan 07 2011 12:08:30 GMT-0500 (EST)

Player

[pausing](#) run

[is not playing](#) run

[is playing](#) run

Song

[prints title](#) run

The screenshot shows the Jasmine Test Runner interface in a web browser. At the top, there's a toolbar with standard browser controls (back, forward, search, etc.) and a tab bar showing various open tabs. The main content area displays the test results. A pink header bar at the top of the results section says "Jasmine 1.0.1 revision 1286311016" and "Show passed skipped". Below this is a green bar showing "1 spec, 0 failures in 0.021s" and the date/time "Finished at Fri Jan 07 2011 12:08:30 GMT-0500 (EST)". The test cases are listed in sections: "Player" and "Song". Under "Player", there are three items: "pausing", "is not playing", and "is playing", each with a "run" button to its right. Under "Song", there is one item: "prints title", also with a "run" button. A red rectangular box highlights the entire "Song" section.

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
------	-----	---------	---------

The screenshot shows a GitHub repository page for 'jazz_money'. The repository has 14 pull requests, 6 issues, and 12 forks. A commit from Mike Grafton on October 30, 2010, was pushed via HTTP. The commit message is: "README update". The commit hash is cfa30a9cd212446d62b6, the tree hash is 80501fc6d0dcfb77095a, and the parent hash is ecf5b73ee40c168c7223.

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

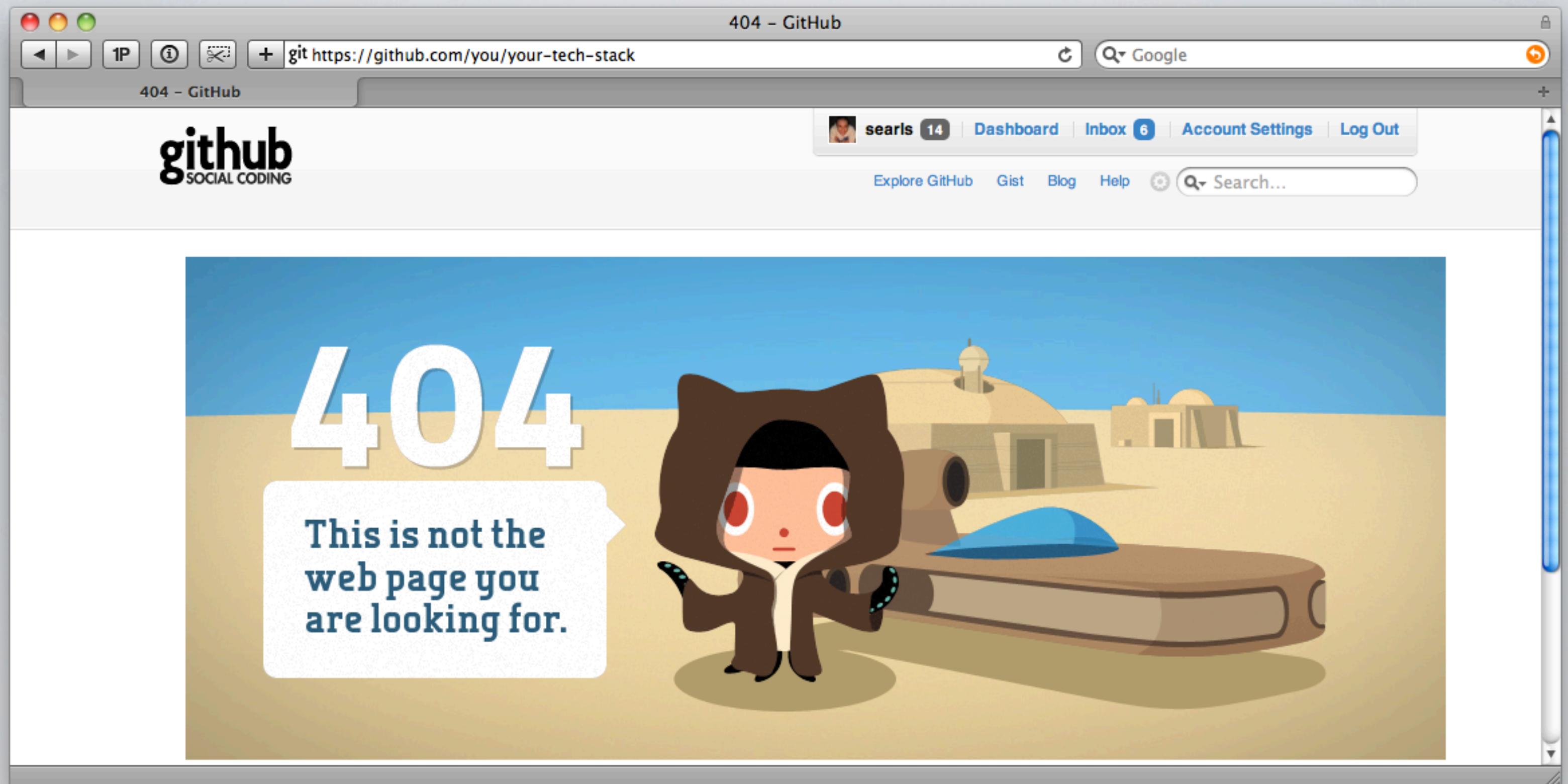
commit 0e8a8ce41eedb1cf318f
tree b8ea48d398d229737dfe
parent 91050e19218c847d09a0

jasmine-maven-plugin /

name age message history

The screenshot shows a GitHub repository page for 'jasmine-maven-plugin' owned by 'searls'. The repository has 23 pull requests and 11 forks. It contains 2 issues and 1 wiki page. The commit history shows a single commit from December 23, 2010, made by 'searls' (author). The commit message is 'doc'. The commit hash is 0e8a8ce41eedb1cf318f, the tree hash is b8ea48d398d229737dfe, and the parent hash is 91050e19218c847d09a0.

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.sxc.hu/photo/1255900>
- ★ <http://www.flickr.com/photos/paperdollimages/512768518/>
(via [*!·sindorella·!*](#))

pillar

@searls

<http://github.com/searls/jasmine-intro>