LEARN FRONTEND TESTING

SEATTLE JS - OCT. 16TH, 2013

Ryan Roemer | @ryan_roemer

@FormidableLabs

SPONSORS

// FORMIDABLE LABS

Coworking for software people.
Opening December 1st in the heart of Fremont.

coworking.formidablelabs.com

REDFIN™

Your New Way Home

MOZ

MENTORS

Try to keep pace with the presentation, but side conversations are encouraged and don't let us stop any good directions.

And, **thanks**!

MOTIVATION

Web applications are increasingly becoming **frontend heavy**.

We need to **verify** app logic and behavior, and that means braving the browser.

SO LET'S TEST

Backend is straightforward and easy

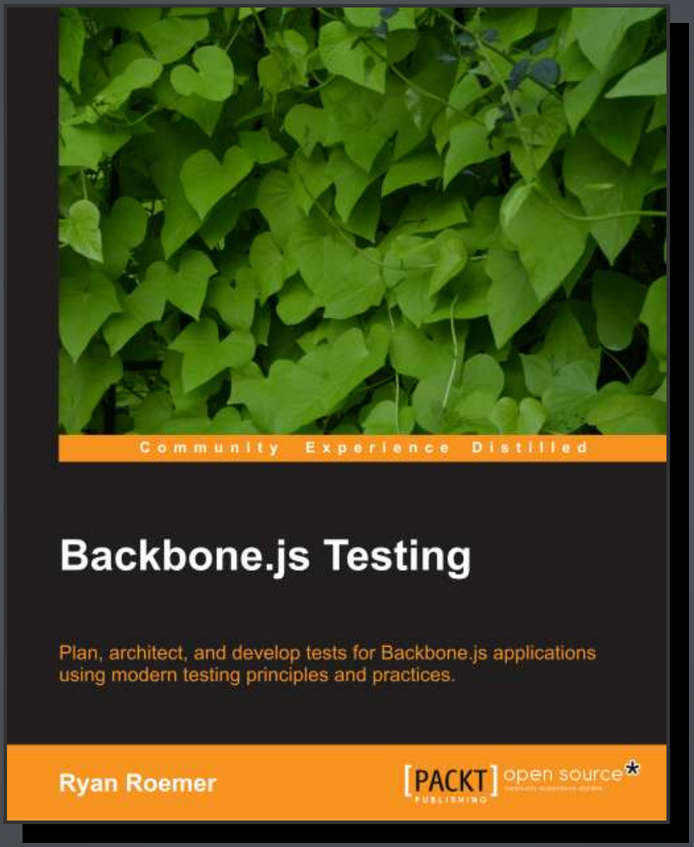
... but what about the frontend?

FRONTEND TESTING

Frontend testing is difficult and error-prone.

- Asynchronous events, timing
- Browser idiosyncracies
- State of testing technologies

BUT GETTING BETTER



Community Experience Distilled

Backbone.js Testing

Plan, architect, and develop tests for Backbone.js applications using modern testing principles and practices.

Ryan Roemer

[PACKT] open source*

... so let's get started with a modern frontend test stack.

GET THE CODE

github.com/FormidableLabs/learn-frontend-testing

```
$ git clone https://github.com/Formidabl
              learn-frontend-testing.git
```

OVERVIEW

- Installation and test page
- Suites
- Assertions
- Fakes
- Automation

WE WILL LEARN HOW TO

- Hook frontend JS to tests
- Write assertions against behavior
- Fake application behavior
- Run and verify the tests

THINGS WE'RE NOT GOING TO COVER

- TDD / BDD
- Application development
- Functional testing
- Performance testing

CODING TIME

We're going to say hello:

"SeattleJS" → "Hello SeattleJS!"

And camel case strings:

"fun-test-time" → "funTestTime"

SET UP YOUR PROJECT

```
# Copy the skeleton application.
$ cp -r skeleton MY_APP_NAME
```

PROJECT STRUCTURE

Using with the "skeleton" application.

```
MY_APP_NAME/
  js/
    app/
      hello.js
      camel.js
    lib/
      chai.js
      mocha.js
      mocha.css
```

HELLO!

skeleton/js/app/hello.js

```
// Hello [VALUE]!
var hello = function (val) {
  return "Hello " + val + "!";
};
```

CAMEL CASE

skeleton/js/app/camel.js

```
// Camel case a string.
var camel = function (val) {
  // Uppercase the first character afte
  return val.replace(/-(.)/g, function
    return first.toUpperCase();
  });
};
```

DEMO

skeleton/index.html

TEST HARNESS

TEST LIBRARIES

- **Mocha** (`v1.13.0`): Framework
- **Chai** (`v1.7.3`): Assertions
- **Sinon.JS** (`v1.8.1`): Fakes - spies and stubs

DIRECTORY LAYOUT

A combined structure.

```
MY_APP_NAME/
  js/
    app/
    lib/
    spec/
      hello.spec.js
      *.spec.js
  test.html
  index.html
```

THE TEST PAGE

Create a test "driver" web page.

example/test.html

```
$ touch MY_APP_NAME/test.html
```

TEST.HTML

```
<html>
  <head>
    <title>Frontend Testing</title>
    <!-- Application libraries. -->
    <script src="js/app/hello.js"></scri
    <script src="js/app/camel.js"></scri
    <!-- Test styles and libraries. -->
    <link rel="stylesheet"
          href="js/lib/mocha.css" />
```

TEST.HTML

```
<!-- Test Setup -->
<script>
  // Set up Chai and Mocha.
  window.expect = chai.expect;
  mocha.setup("bdd");

  // Run tests on window load.
  window.onload = function () {
    mocha.run();
```

TEST.HTML

```
    <!-- Tests. -->
    <!-- ... spec script includes go her
  </head>
  <body>
    <div id="mocha"></div>
  </body>
</html>
```

example/test-empty.html

MOCHA SUITES, SPECS

- Spec: A test.
- Suite: A collection of specs or suites.

SUITES, SPECS

test-mocha.html | mocha-suite.spec.js

```
describe("single level", function () {
  it("should test something");
});

describe("top-level", function () {
  describe("nested", function () {
    it("is slow and async", function (do
      setTimeout(function () { done(); 
    });
```

SETUP, TEARDOWN

test-mocha.html | mocha-setup.spec.js

```
describe("setup/teardown", function () 
  before(function (done) { done(); });
  beforeEach(function () {});

  after(function (done) { done(); });
  afterEach(function () {});

  it("should test something");
});
```

CHAI ASSERTIONS

- Natural language syntax.
- Chained assertions.

CHAI API

The **"bdd" API**:

- **Chains**: `to`, `be`, `been`, `have`
- **Groups**: `and`
- **Basics**: `a`, `equal`, `length`, `match`

HELLO!

test-hello.html | hello.spec.js

```
describe("hello", function () {
  it("should say hello", function () {
    expect(hello("World"))
      .to.be.a("string").and
      .to.equal("Hello World!").and
      .to.have.length(12).and
      .to.match(/He[l]{2}/);
  });
});
```

CAMEL CASE

test-camel.html | camel.spec.js

```
describe("camel", function () {
  it("handles base cases", function () {
    expect(camel("")).to.equal("");
    expect(camel("single")).to.equal("si
  });
  it("handles dashed cases", function ()
    expect(camel("a-b-c")).to.equal("aB
    expect(camel("one-two")).to.equal("
  });
```

MORE CHAI

test-chai.html | chai.spec.js | chai-fail.spec.js

```
describe("chai", function () {
  it("asserts", function () {
    expect(["one", "two"]).to.contain("t
    expect({foo: {bar: 12}})
      .to.have.deep.property("foo.bar",
  });
});
describe("chai", function () {
  it("fails", function () {
```

SINON.JS FAKES

Dependencies, complexities? Fake it!

- **Spies**: *Observe* function behavior.
- **Stubs**: *Spies* that *replace* behavior.
- **Fake Server**: Override `$.ajax`, etc.

SINON.JS SPY

test-sinon.html | camel-spy.spec.js | camel.js

```
describe("camel", function () {
  it("spies upper case", function () {
    var spy = sinon.spy(String.prototype

    expect(spy.callCount).to.equal(0);
    expect(camel("a-b")).to.equal("aB");
    expect(spy.callCount).to.equal(1);
    expect(spy.firstCall.returnValue).to
```

SINON.JS STUB

test-sinon.html | camel-stub.spec.js | camel.js

```
describe("camel", function () {
  it("stubs upper case", function () {
    var stub = sinon.stub(String.prototy
      function () { return "FOO"; });

    expect(camel("a-b")).to.equal("aFOO'
    expect(stub.callCount).to.equal(1);

    stub.restore();
```

AUTOMATION

Drive our frontend tests with PhantomJS using Mocha-PhantomJS

PREP TEST.HTML

Update the test.html file:

```
window.onload = function () {
  (window.mochaPhantomJS || mocha).run(
};
```

HEADLESS!

Install and drive tests from the command line:

```
$ npm install mocha-phantomjs
$ node_modules/.bin/mocha-phantomjs \
  MY_APP_NAME/test.html
```

... and that's all for now!

WHAT WE'VE COVERED

- Test harness
- Suites, specs
- Assertions
- Fakes
- Automation

ADDITIONAL TOPICS

- Advanced testing: DOM, fixtures
- TDD / BDD
- Functional testing
- Performance testing
- Continuous Integration: (Travis CI)

THANKS!

Ryan Roemer | @ryan_roemer

bit.ly/frontend-testing
bit.ly/frontend-testing-src
backbone-testing.com