Testing

Liam McLennan

@liammclennan

Outline

- The benefits of testing JavaScript applications
- Testing Tools
  - Jasmine
- Testing models
- Testing views
- Testing without a browser

Reasons to Test

- **To catch bugs**
  - As a dynamic language, JavaScript will not report problems at compile time
- **To enable change**
  - Without comprehensive tests change is extremely difficult
- **To account for browser differences**
  - Automated tests will help you to keep your application running across all supported browsers

Testing Tools

- **Test runner**
  - Jasmine
  - Mocha
  - QUnit

Jasmine

context

spec

```
describe('some context', function () {
  it('should show some observable behavior', function () {
      // assert expectations here
  });
});
```

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

Jasmine

```
context   describe('some context', function () {
context     describe('nested context', function () {
spec          it('should show some observable behavior', function () {
                // assert expectations here
              });
spec          it('should show some other behavior', function () {
                // assert expectations here
              });
            });
          });
```

Testing Models

- **Testing models is easy!**

Test Pattern

```
1. Initialize a model with a specific state
2. Test that the model’s behavior matches
   expectations
3. Goto 1
```

Rectangle Model

- A rectangle has a length and a width

Rectangle Specification

```
Rectangle
1  with length 7 and width 4
     should have an area of 28
2    should have a perimeter of 22
```

Test Pattern

```
1. Initialize a model with a specific state
2. Test that the model’s behavior matches
   expectations
3. Goto 1
```

Testing Models (cont.)

Rectangle Specification

context

```
Rectangle
```

Jasmine Specification

context

```
describe('Rectangle', function () {











});
```

Testing Models (cont.)

Rectangle Specification

context

```
Rectangle
  with length 7 and width 4
```

Jasmine Specification

context

```
describe('Rectangle', function () {
  describe('with length 7 and width 4', function () {







  });
});
```

Testing Models (cont.)

Rectangle Specification

spec

```
Rectangle
  with length 7 and width 4
    should have an area of 28
```

Jasmine Specification

spec

```
describe('Rectangle', function () {
  describe('with length 7 and width 4', function () {
    it('should have an area of 28', function () {
      // assert expectations here
    });



  });
});
```

Testing Models (cont.)

Rectangle Specification

spec

```
Rectangle
  with length 7 and width 4
    should have an area of 28
    should have a perimeter of 22
```

Jasmine Specification

spec

```
describe('Rectangle', function () {
  describe('with length 7 and width 4', function () {
    it('should have an area of 28', function () {
      // assert expectations here
    });
    it('should have a perimeter of 22', function () {
      // assert expectations here
    });
  });
});
```

Full Rectangle Specification

Rectangle Specification

```
Rectangle
  with length 7 and width 4
    should have an area of 28
    should have a perimeter of 22
  with equal length and width
    should be a square
  with unequal length and width
    should not be a square
  setting invalid values
    negative length or width
      should throw an error
    zero length or width
      should throw an error
```

Testing Views

- **Write testable views!**
  - Do not depend on specific DOM elements
  - Render a completely new DOM element for the view or render into an element passed to the views constructor.
  - Never access DOM elements outside of the view
- **Test**
  - Rendered elements
  - Raised events

The Rectangle View

Rectangle View Specification

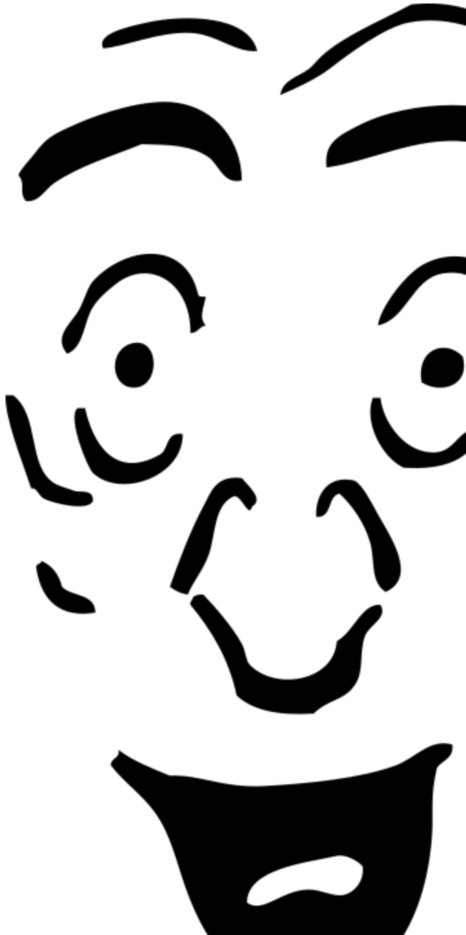
```
Rectangle View
  with length 70 and width 40
    should render a div with class rectangle
    should have dimensions 70 x 40
    should raise rectangle:selected when clicked
```

Testing Routes

- Don’t
- Keep all logic out of route handlers so that route handlers aren’t required for testing.

Testing without a browser

- **Jasmine-node or Mocha**
  - For tests that do not require a DOM
- **Use a headless browser (phantom.js)**
  - Tests in a full browser environment



Jasmine-Node

- Run jasmine tests on node.js
- https://github.com/mhevery/jasmine-node

Phantom.js

- Headless webkit with JavaScript API
- Useful for running tests in a browser from the command line
- http://phantomjs.org/
- https://github.com/ariya/phantomjs/

Summary

- Testing client-side JavaScript is tricky
- Testing improves velocity
- Model testing
- View testing
- Router testing
- Browser-less testing