Testing Javascript applications

Irina Dumitrascu

ruby & cofeescript | dira.ro , @dira_geek_girl

23 August 2012

Why?

Well…

And…

Every time you hit refresh & click to test…

And…

Every time you hit refresh & click to test…

God kills a kitten.

And…

Every time you hit refresh & click to test…

God kills a kitten. (image not available)

Without refresh...

Without refresh...

We're at the mercy of the machines.

Show me the code!

Show me the code!

no code to show :(

Show me the code!

no code to show :(

let's make a new app! :)

Show me the code!

no code to show :(

let's make a new app! :)

what will I write the presentation in?

let's write some code for showing presentations!!

Show me the code!

no code to show :(

let's make a new app! :)

what will I write the presentation in?

let's write some code for showing presentations!!

> Are you crazy?

Show me the code!

no code to show :(

let's make a new app! :)

what will I write the presentation in?

let's write some code for showing presentations!!

Are you crazy?

or

Do you hate sleeping?

How it works

- the presentation is written in a Markdown file
- the Markdown is parsed into HTML
  - `h1` is the start of a new slide
  - `hr` is the start of my notes (hidden)
- and enhanced with `next` and `previous`

The fascinating part

I wrote all this functionality test-first.

The fascinating part

I wrote all this functionality test-first.

Without seeing it in the browser until it was done.

The fascinating part

I wrote all this functionality test-first.

Without seeing it in the browser until it was done.

And it worked.

The fascinating part

I wrote all this functionality test-first.

Without seeing it in the browser until it was done.

And it worked.

(Because it was tested.)

The fascinating part

The result is the presentation that you see right now.

So let's talk about testing

Kinds of testing

Unit testing

- Does this **function** behave properly?

Integration testing

- Does the **application** behave properly?

Unit testing

- tests independent functions
- fast and focused
- makes sure independent pieces work

How to do it

1. think about what you must implement
2. break it in small functions

How to do it

1. think about what you must implement
2. break it in small functions
3. choose a function and write a test for it. watch it fail.

How to do it

1. think about what you must implement
2. break it in small functions
3. choose a function and write a test for it. watch it fail.
4. write the code that fixes the test

How to do it

1. think about what you must implement
2. break it in small functions
3. choose a function and write a test for it. watch it fail.
4. write the code that fixes the test
    - check for refactoring opportunities

How does it look like?

Let's see a simple test.

Unit testing

You need to:

- separate your JS from the DOM (no inline JS)
- structure the code (classes, modules)
- separate the definition and the running step (document.ready …)
- the smaller the thing to test, the better

Unit testing - don'ts

- do not test for the same thing in multiple tests
- do not test for texts appearing in the DOM (test for classes and ids)

Unit testing - do's

- test the public interface of each object
- test for classes and ids (not strings) in the UI
- mock out dependecies (HTTP requests & HTML in the page, but also collaborators)
- maintain the tests as you do with the code

Many tools

Frameworks

jasmine, mocha, buster.js

Separate assertion libraries

chai

Tools

to run the tests from the command line, to integrate with CI

Added benefits

- better structured code

And…

But…

- the individual pieces work
- there's no guarantee that they work well together!

Integration testing

- is the application as a whole working as expected?
  - also called acceptance testing

Integration testing

- create scenarios and test that they work
- have to setup the whole application stack to a known state

Integration testing

- slow
- do not know about the implementation of individual pieces

Conclusion

- test your code!
- unit tests and integration tests complement each other

Conclusion

- test your code!
- unit tests and integration tests complement each other
- **think of the kittens!**

Starring

Tests

- mocha as *Test framework*
- chai as *Assertive guy*
- sinon as *Mocking and spying specialist*

Code

- jQuery
- Rainbow as *Code stylist*
- markdown css as *Text stylist*

Presentation

- http://www.flickr.com/photos/kaibara/4068996309/ as *Test Ninja Cat*

Check out the code

https://github.com/dira/js-testing-presentation

Thank you!

Questions?