Zero to Testing in JavaScript

Basics of testing, in JS

About me

- Blog: thewebivore.com
- Twitter: @pamasaur
- Co-organizer of Philadelphia JavaScript Developers
- Podcasting on Turing Incomplete (@turingcool)
- Testing fanatic

Welcome to the testing track?

pause … A TESTING TRACK!

Let’s kick this off right!

Agenda

- My testing story
- Testing theory
- Basic test walk-through
- Testing frameworks
- Other forms of testing
- Working testing into your workflow
- Minishop (workshop)

My testing story

When I first started writing tests

Code Retreat

- TDD
- Pairing
- Throw-away code

What about tests in *my* projects?

previously:

“Tests* are doing the dishes”

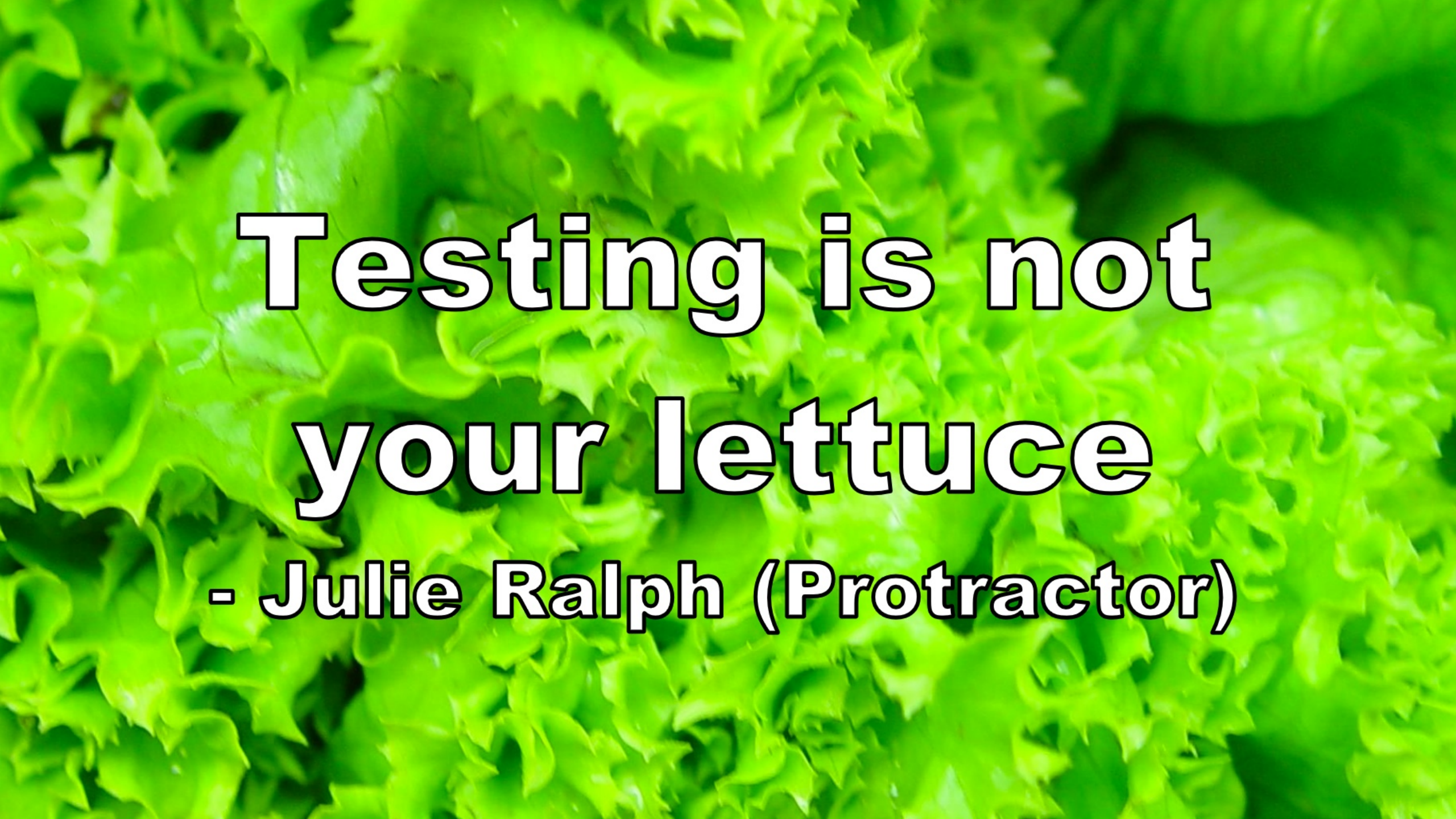
* and writing docs and fixing bad code

aka:

“Tests are work for people you don’t like”

The other piece that I would add to this is, and I've said this before at talks, that testing isn't something that you do because it is good to be testing. It's not your lettuce. You do it because it increases your confidence in your code. And it increases your ability to release quickly and make changes without worrying that something is going to break. And so, if you're paranoid like me, that's a good incentive to do testing. And if you have a whole lot of self-confidence, maybe you're just a really great coder. But maybe it's a little bit more likely that you're going to make mistakes and you might pay for it a little bit later.

Julie Ralph, Protractor
http://javascriptjabber.com/106-jsj-protractor-with-julie-ralph/


Testing is not your lettuce
- Julie Ralph (Protractor)

now:

GOOD developers write tests

Literally.

`-b` for bad coder

What do you call code without tests?

Legacy code.

Testing basics

1. When you break it, it breaks!
2. Secondary line of documentation defense
3. Design tool

When you break it, it breaks!

- Testing is required to enable refactoring
- Regularly run tests
- Use test results to make decisions

Documentation defense

- Tests describe behavior
- Read tests to understand behavior
- Write tests to describe behavior
- Write tests to validate behavior

Tests as design tool

- Tests describe functionality
- Write functionality description (test)
- Write code to make it work

BDD/TDD

- Behavior Driven Development
- Test-Driven Development

"Write tests to inform how you write your code"

BDD

- describe
- it
- before
- after
- beforeEach
- afterEach

TDD

- suite
- test
- setup
- teardown

Test Ever vs. Test Never

Types of testing

- Unit
- Integration
- Functional
- E2E

Unit

Testing functionality in isolation

"When I call addNumbers, it returns the value of the added numbers"

Unit testing takeaway:

Test *your* code

How to only test your code?

- Isolate
- Faking

Fake things: Spies, stubs, and mocks

- Spy: an object that records its interactions
- Stubs: fake objects
- Mocks: fake objects with expected behavior

Generally, you can SPY on a function, STUB an object, and MOCK a service.

Integration

- Multiple pieces of code together

"Is the router setting up my models correctly?"

Functional

- Does your code work as product expects it to? (functional requirements)

"When I click the login button, it should log me in"

E2E (end-to-end)

- Functional, but fully integrated with external services, simulate real-time situations.

"When I click 'Login with Google,' it logs me in"

JavaScript test walk-through

JavaScript test walk-through

- Setting up your base
- Writing the test/expectations*
- Making it pass*

* The order of these is debatable

Jasmine

- jasmine.github.io

- Download a standalone version on GitHub from pivotal/jasmine
- Node: jasmine-node (fork of karma-jasmine)

Mocha

- visionmedia.github.io/mocha

- Node!
- `npm install –g mocha`

Anatomy of a test

Describe [thing you're testing]

It [does something you expect it to do]

Rinse and repeat.

Example test walk-through

with Mocha

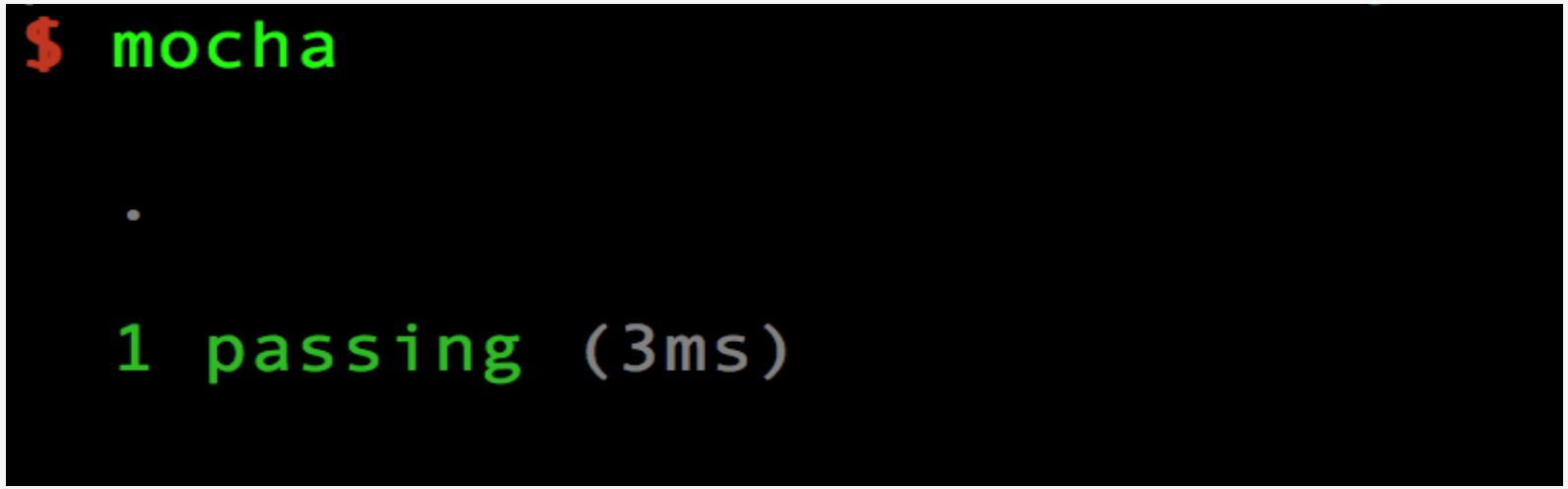
```
var assert = require('assert');
describe("An area of my application", function() {
    it("should know that 2 and 2 is 4", function(){
        assert.equal(4, 2+2);
    });
});
```

```
var assert = require('assert');
describe("An area of my application", function() {
    it("should know that 2 and 2 is 4", function(){
        assert.equal(4, 2+2);
    });
});
```

```
var assert = require('assert');
describe("An area of my application", function() {
    it("should know that 2 and 2 is 4", function(){
        assert.equal(4, 2+2);
    });
});
```

```
var assert = require('assert');
describe("An area of my application", function() {
    it("should know that 2 and 2 is 4", function(){
        assert.equal(4, 2+2);
    });
});
```

```
var assert = require('assert');
describe("An area of my application", function() {
    it("should know that 2 and 2 is 4", function(){
        assert.equal(2+2, 4);
    });
});
```


$ mocha
.
1 passing (3ms)

Testing Tools

Test describers

- Jasmine
- Mocha
- QUnit
- node-tap
- YUI Test

Assertions

- Chai
- should.js
- Expect.js
- better-assert

Spies, stubs, and mocks

- Sinon.js
- Jest from Facebook

Test runners

- Karma
- Testem
- YUI Yeti

Bringing testing into the fold

3 tips for making testing a regular part of your world

#1: Teach testing

- Attend talks like this!
- Practice (ex. Code Retreat)
- Pair programming

#2: Code coverage

- Istanbul
- Blanket.js

#3: Code review

- Quality assurance
- Mentoring
- Don’t accept without tests!

What'd we learn?

- Basics of testing
- Writing a JavaScript test
- Tools in JavaScript for testing
- Ways to create a testing culture

Minishop

(workshop, playshop …)

TODO

- Set up karma (our test runner)
- With Jasmine (our assertion framework)

TODO

- Write a failing test
- Correct the failing test

TODO

- Write another test
- Setup in beforeEach

Thank you!

- Find me online at
  - @pamasaur
  - thewebivore.com (blog)
  - turing.cool (podcast)
  - thewebivore.com/book (book!)