AUTOMATED TESTING

HOW TO KNOW YOUR CODE WORKS

Created by Chris Foster / @chrisfosterelli

WHAT IS TESTING?

AUTOMATED VS. MANUAL

- Manual testing: a human tests the code
- Automated testing: code tests the code

AUTOMATED TESTING

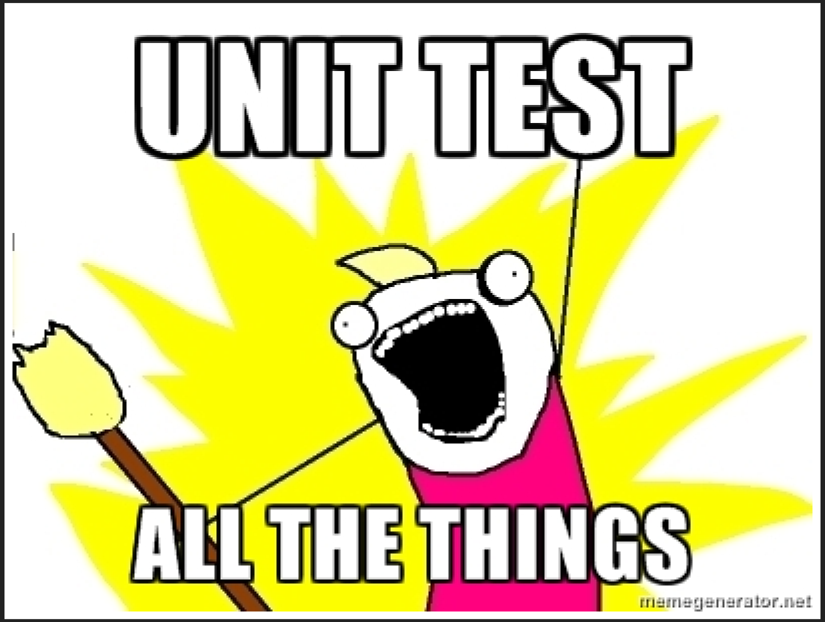
1. Unit testing
2. End-to-end

We're going to learn about Unit testing!

WHY UNIT TEST?

- Reliable code
- Modular code
- Maintainable code
- Error resistant code

UNIT TESTING



In unit testing the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation.

JS TESTING

- Mocha
- Chai
- Jasmine
- Sinon
- Qunit
- expect.js
- and more...

- Better-assert
- Unexpected
- Jest
- JSUnit
- UnitJS
- should.js
- and more...

EXAMPLE UNIT

```javascript
function getGreeting(name, greeting) {
  if (!greeting) greeting = 'Hello ';
  if (name) return greeting + name + '!';
  return greeting + '!';
}
```

What do we need to test for?

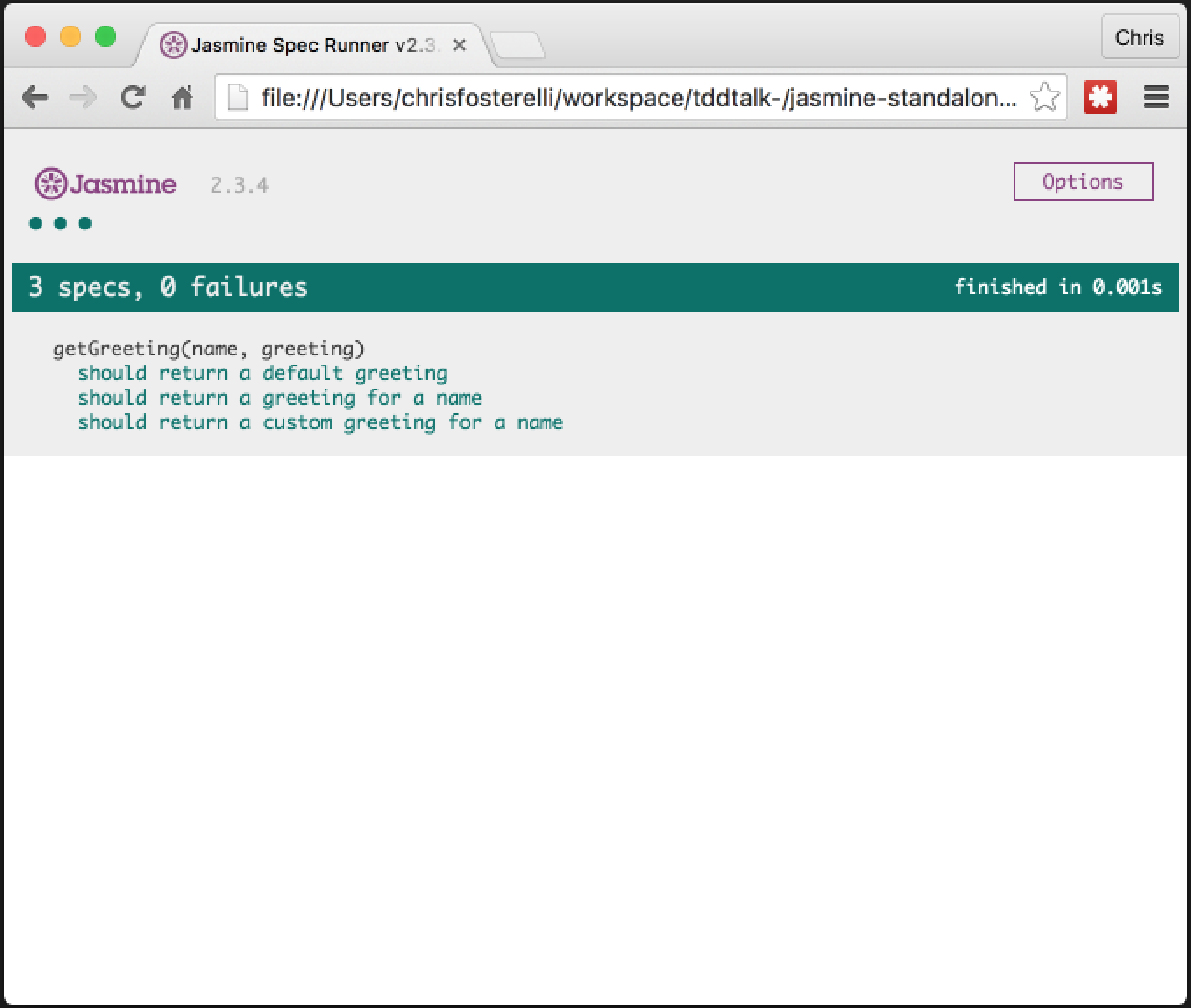
TEST CASES

1. Given no name or greeting -> returns 'Hello!'
2. Given only a name 'John' -> returns 'Hello John!'
3. Given 'John' and 'Good day ' -> returns 'Good day John!'

TEST SUITE

```
describe('getGreeting(name, greeting)', function() {

  it('should return a default greeting', function() {
    expect(getGreeting()).toEqual('Hello!');
  });

  it('should return a greeting for a name', function() {
    expect(getGreeting('Sam')).toEqual('Hello Sam!');
  });

  it('should return a custom greeting for a name', function() {
    expect(getGreeting('Sam', 'Good Day')).toEqual('Good Day Sam!');
  });

});
```

SUCCESS!


Jasmine Spec Runner v2.3
Chris
file:///Users/chrisfosterelli/workspace/tddtalk-/jasmine-standalon...
Jasmine
2.3.4
Options

```
3 specs, 0 failures                                    finished in 0.001s

getGreeting(name, greeting)
  should return a default greeting
  should return a greeting for a name
  should return a custom greeting for a name
```

LET'S WRITE SOME TESTS!

http://bit.ly/1I24eLn

PICKING UNITS

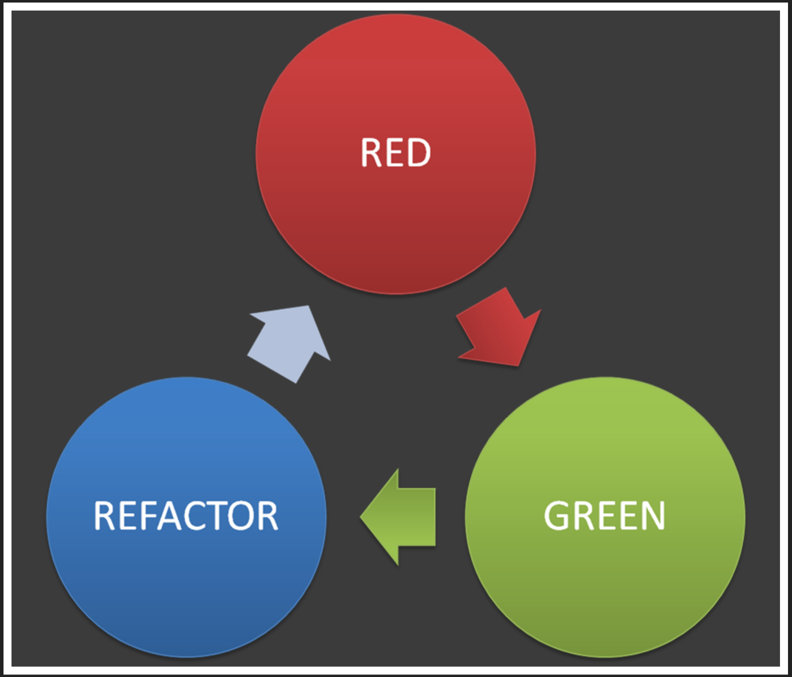
- Java: class
- Node: module
- Browser: file

TESTING WORKFLOW

- Code is only accepted if tests written
- Code is only accepted if tests pass
- Green builds and Red builds

TEST DRIVEN DEVELOPMENT

- Write tests *before* code
- Emphasis on minimal code
- Emphasis on *code coverage*



SPIES, STUBS, AND MOCKS

- A **mock** is a fake object for testing upon
- A **stub** is a function with pre-programmed behaviour
- A **spy** is a 'wrapper' around another function

FIXTURES

- Refreshed during each test
- Fake test data loaded into a database
- Allows your tests to remain reliable

FAKING IT

- System clock
- HTTP requests
- Entire modules
- Database values

DEALING WITH ASYNC

```
function setAndWait(cb) {
  setTimeout(function() {
    window.global = true;
    cb();
  }, 5000);
}
```

```
it('should call the timeout', function(done) {
  window.global = false;
  setAndWait(function() {
    expect(window.global).toEqual(true);
    done();
  });
});
```

SETUP

```
describe('My Suite', function() {
  var database, foo;

  before(function() {
    foo = 'static value';
  });

  beforeEach(function() {
    database = new Database();
  });

  /* [... tests ...] */

});
```

TEARDOWN

```
describe('My Suite', function() {
  var database, foo;

  /* [... tests ...] */

  afterEach(function() {
    database.close();
  });

  after(function() {
    console.log('All tests done!');
  });

});
```

NESTING DESCRIBE

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

  describe('when the server is down', function() {
    it('should return an error', /*...*/);
    it('should close the connection', /*...*/);
  });

  describe('when the server is up', function() {
    it('should return the connection', /*...*/);
  });

});
```

MORE ADVANCED TESTS!

http://bit.ly/1I2AkGI

THE END

- @chrisfosterelli
- https://fosterelli.co