Testing in AngularJS

Pete Bacon Darwin

Why Test?

- External Drivers
  - Contractual requirement
  - User confidence
  - **Increase project velocity!**
- Internal Drivers
  - Because developers are not gods!
  - **Encourages good design!**
  - Freedom to evolve / refactor

Why Don’t People Test?

- Not Enough **Time**
  - Belief that writing tests increases effort
- No Defined Test **Strategy**
  - Don’t know where to start
- Poor Test **Tools**
  - Difficult to automate testing tasks
- App / **Framework** is not test-friendly
  - High Coupling or Low Cohesion

What to Test?

- Application Code
  - Isolated **Unit Tests**
- Whole Application
  - Functional **End 2 End Tests**

How *not* to Test?

- Manual ad hoc Testing
  - Debugger / Console
  - Navigating around the App
  - Slow / Time Consuming
  - Boring / Error Prone
  - Unlikely to Catch Regressions

Automate Testing

- Tests: **Write Once - Run Often**
  - Prevent regressions
- Run Tests on **Every Save**
  - Rapid feedback on bad code
- **Continuous** Integration
  - Keep developers honest

AngularJS and Testing

http://docs.angularjs.org/guide

Misko Hevery - likes testing

http://misko.hevery.com/

- From Misko's blog...
  - How to get started with **TDD**
  - Design for **Testability**
  - Cost of **Testing**
  - Psychology of **Testing** at Wealthfront Engineering
  - RTAC2010 - All hands on **Testing**
  - Growing Object Oriented Software, Guided By **Tests**
  - There are lots of ways to **Test** your JavaScript, …

Angular Does Testing

- Application Structure
  - Separation of Concerns
    - Data Binding
    - Dependency Injection
- Test Helpers
  - Mock Services
  - Helper Functions
  - Synchronized Tests
- Tooling Support
  - Karma Test Runner
  - Protractor E2E Test Runner

Unit Testing

http://docs.angularjs.org/guide/dev_guide.unit-testing

Unit Tests in AngularJS

Write tests in...

http://pivotal.github.io/jasmine/

Run tests with...

http://karma-runner.github.io/

Jasmine Test Specs

```
describe ‘my dog’, ->
  it ‘should bark at intruders’, ->
    spyOn(myDog, ‘bark’)
      myDog.see(intruder)
      expect(myDog.bark).toHaveBeenCalled()
```

Warning! CoffeeScript

Jasmine Helpers

- `describe() / it()`
  - Describe what the code should do
- `xdescribe() / xit()`
  - Don’t run this describe or it block
- `beforeEach() / afterEach()`
  - run this block before/after every test in this describe
- `expect() / spyOn()`
  - check a value or whether a function was called

Jasmine Matchers

- `toEqual()`, `toBe()`
- `toBeGreaterThan()`, `toBeLessThan()`
- `toBeTruthy()`, `toBeFalsy()`
- `toMatch()`, `toContain()`
- `toBeNull()`, `toBeDefined()`
- `toThrow()`, …
- `...not...`

Jasmine in a Browser

- In-browser Test Runner

```
<link rel="stylesheet" href="http://cdn.jsdelivr.net/jasmine/1.3.1/jasmine.css" />
<script src="http://cdn.jsdelivr.net/jasmine/1.3.1/jasmine.js"></script>
<script src="http://cdn.jsdelivr.net/jasmine/1.3.1/jasmine-html.js"></script>
```

- Run Jasmine

```
var jasmineEnv = jasmine.getEnv();
jasmineEnv.addReporter(new jasmine.HtmlReporter());
jasmineEnv.execute();
```

Demo Code

https://github.com/petebacondarwin/angularjs-testing-presentation

Demo - `average()` Unit Tests

- average function
  - should return the average of the values in an array
  - should return 0 for an empty array
  - should throw invalid parameter `values`, if not an array

AngularJS Test Helpers

angular-mocks.js

- `module('moduleName')`
  - load a **module** to be tested
- `inject(function(injectables) {...})`
  - get services from the **injector**
- `ddescribe() / iit()`
  - only run this **describe** or **it** block

Loading Angular Modules

- Can be called “**before each**” test
  - `beforeEach(module(‘module1’, ‘module2’));`
- Useful for loading mock code
  - `module(‘myApp’, ‘myMocks’);`

Dependency Injection

- Inject Angular services directly into tests

```
describe('myService', function() {
  it('should return "X", inject(function(myService) {
    expect(myService()).toEqual("X");
  }));
});
```

Caching Injected Services

- Share a service between tests

```
describe('myService', function() {
  var myService;
  beforeEach(inject(function(_myService_) {
    myService = _myService_;
  }));
});
```

Testing Angular Services

- **Load** the module
- **Inject** the service
- Set up **test data**
- **Use** the service
- Check the **expected** result

Demo - average() Angular Service

```
angular.module('average-app', [])
  .factory('average', function() {
    return function average(values) {
      …
    };
  });
```

Testing Angular Filters

- Filters are just Angular Services
  - with the word “Filter” added to their name
- Inject the filter directly:

```
inject(function(limitCharsFilter) { … });
```

- Or use $filter service:

```
inject(function($filter) {
  var limitCharsFilter = $filter(‘limitChars’);
}
```

Testing Angular Controllers

- Use the **$controller** service to get an instance of your controller

```
inject(function($controller, $rootScope) {
  var myCtrl = $controller('MyCtrl, {
    $scope: $rootScope
  });
  ...
});
```

Demo - limitChars

- Test a **controller** binding a service to a view
  - **Instantiate** the controller
  - **Spy** on the service
  - **Use** the controller
  - Check whether the **spy was called**

Testing AngularJS Directives

- **Compile** HTML containing the directive

```
element = $compile('<div my-directive></div>')($scope);
```

- Interact with the **`element`**:

```
element.val('new value');
```

- Modify the **`$scope`**:

```
$scope.prop = 'other value';
$scope.$digest();
```

Demo - button directive

- Twitter Bootstrap **button** formatting

```
<button type="submit" size="large">Press Me!</button>
```

- Automatically add CSS classes

```
<button class="btn btn-primary btn-large"
  type="submit" size="large">Press Me!</button>
```

Handling Asynchronous Code

- Testing async code is **problematic**
  - Async handler executed in a **different call-stack**
  - **Wait for callback** to occur - slow tests
  - Rely on **external resources** - non-deterministic
- Replace with **synchronous** code!
  - **Mock** the code that calls handlers asynch
  - With code that calls handlers **synchronously**

AngularJS and Async Mocks

- angular-mocks.js provides **mock services**
  - `$timeout`
  - `$interval`
  - `$httpBackend`
- Includes **synchronous flush** mechanisms
  - `$httpBackend.flush();`
  - `$timeout.flush();`

ngMock.$timeout

- Timeouts are added to a flushable queue

```
doSomething = jasmine.createSpy();
$timeout(doSomething, 100);
$timeout.flush();
expect(doSomething).toHaveBeenCalled();
```

ngMock.$httpBackend

- Mock up responses to requests and flush

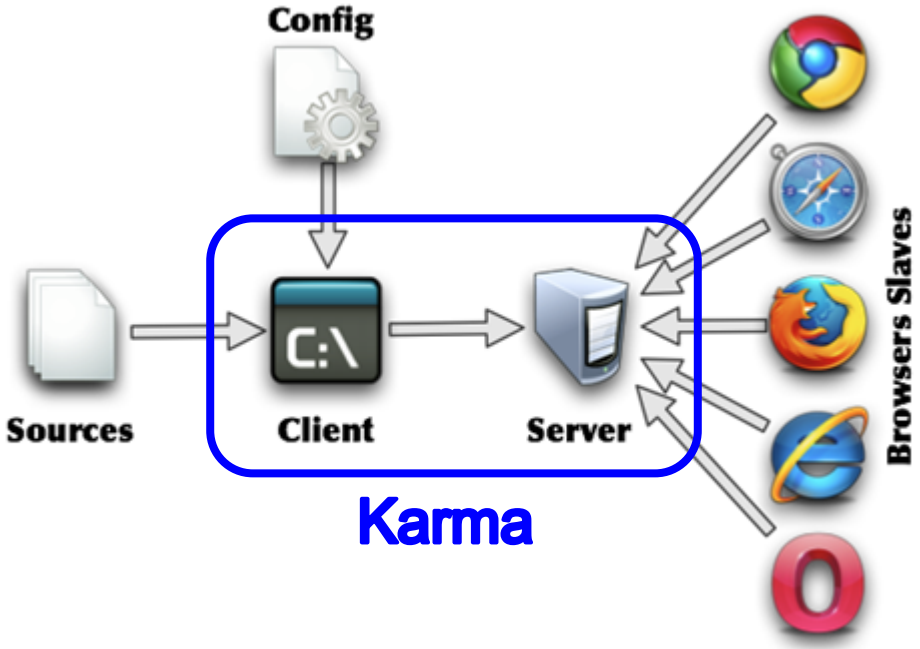
```
$httpBackend.expectGET(‘/some/url/’)
    .respond({ some: ‘thing’ });

$http(‘/some/url/’).success(function(data) {
  $scope.data = data;
});
$httpBackend.flush();

expect($scope.data).toEqual({ some: ‘thing’ });
$httpBackend.verifyNoOutstandingExpectation();
$httpBackend.verifyNoOutstandingRequest();
```

Running Unit Tests

http://karma-runner.github.io

Running Unit Tests

- **Karma** Runs Your Unit Tests in Browsers



Using Karma

- Install as a global npm module
  - `npm install -g karma`
  - `karma init`
- Don’t forget to reference src, lib & spec files
  - `angular.js, angular-mocks.js`
  - `src/*.js, test/*.js`
- `karma start`

```
$ karma start
INFO [karma]: Karma v0.10.6 server started at http://localhost:9876/
INFO [launcher]: Starting browser Chrome
INFO [Chrome 31.0.1650 (Windows 7)]: Connected on socket g4nQy-JyGTdQIajHHEus
Chrome 31.0.1650 (Windows 7): Executed 2 of 2 SUCCESS (2.863 secs / 0.016 secs)
```

End to End Tests

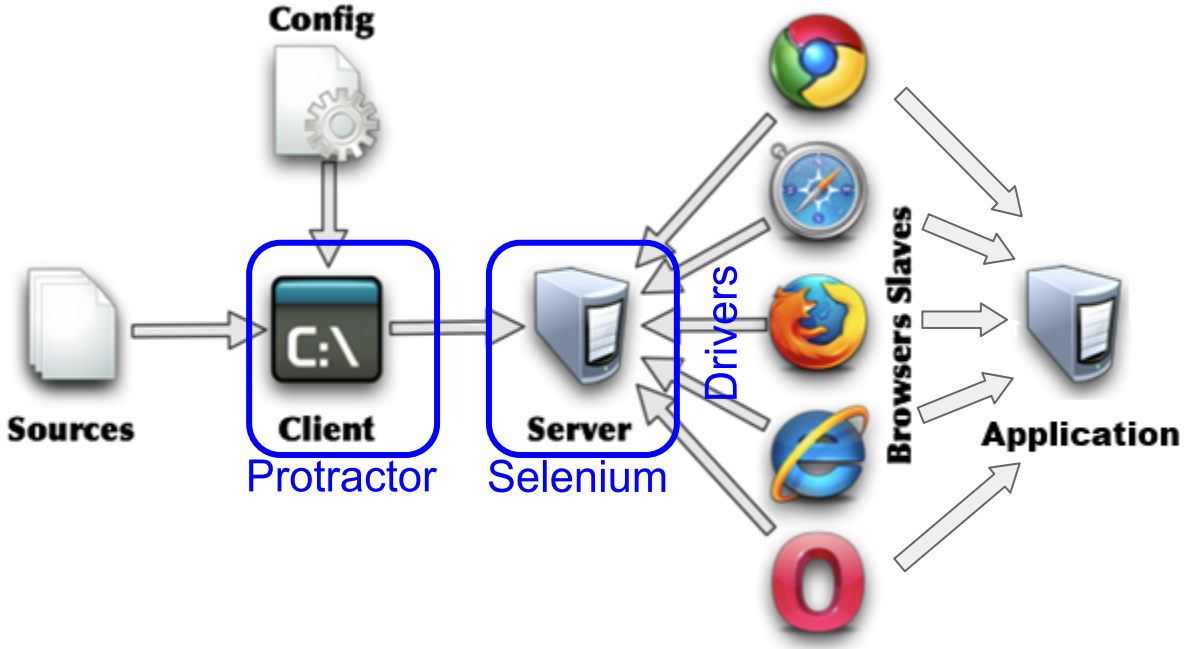
https://github.com/angular/protractor

Functional / End to End Testing

- Test whole application from front to back
- **Protractor** - browser automation framework
  - Angular specific tool
  - Wraps **Selenium-WebDriverJS**
- Similar but **different** to Karma
  - Run against a **full instance** of the app
  - Tests are out of band and **async**

Protractor Architecture

- Stand-alone Selenium Server + Drivers



Setting up Protractor

- Install npm module
  - `npm install -g protractor`
- Download selenium and chrome driver
  - `webdriver-manager update`
- Create a config file
  - How to connect to selenium?
  - What browser drivers to use?
  - What specs to run?

Protractor Test Helpers

- There are five global helpers
  - `protractor` - access to webdriver
  - `browser` - an instance of protractor
  - `element()` - function for element location
  - `by.<strategy>()` - used when locating elements
  - `$()` - alias for `element(by.css())`

Page Object DSLs

- You can create your own test DSL

```
var angularHomepage = {
  nameInput: element(by.model('yourName')),
  greeting: element(by.binding('yourName')),
  get: function() {
    browser.get('http://www.angularjs.org');
  },
  setName: function(name) {
    angularHomepage.nameInput.sendKeys(name);
  }
};
```

Using a Page Object DSLs

- DSLs can make tests cleaner

```
describe('angularjs homepage', function() {
  it('should greet the named user', function() {
    angularHomepage.get();
    angularHomepage.setName('Julie');
    expect(angularHomepage.greeting.getText())
      .toEqual('Hello Julie!');
  });
});
```

Demo - Protractor

- Test AngularJS home page
  - http://angularjs.org

Round Up

- Do **write unit tests** and **functional tests**
- **AngularJS helps** with testing
  - Dependency injection
  - Mock services
  - Integration with tools
- Run unit tests automatically with **Karma**
- Run functional tests with **Protractor**

Buy my Book!

Packed with real solutions to real problems!

(There is a whole section on unit testing)

http://www.packtpub.com/angularjs-web-application-development/book