Unit Testing Angular Like a Boss

Prerequisites

- Get hold of a TypeScript code editor (e.g. Visual Studio Code or WebStorm)
  - Install Visual Studio Code
- Install Node.js (maybe nvm too)
  - Install node 8.9.4 or later and npm 5.6.0 or later
  - Or use nvm
- Clone the repository
  - `git clone https://github.com/joeeames/UnitTestingAngularLikeABoss.git`
  - `cd angular2-testing`
- Install the angular CLI tool globally (needs to be >= beta.15)
  - `npm install -g @angular/cli`
- The day before the workshop run “git pull” to update to the latest version of the repo since there may be changes

Browse the full Prerequisites document with more detailed information and instructions.

Overview

Step 1: Introduction

Resources

- Arrange - Act - Assert
- Unit Testing: Mocks, Stubs and Spies
- http://martinfowler.com/articles/mocksArentStubs.html
- Jasmine Spies - API reference

Presentation

- Introduction through Unit Testing in Angular

Demonstration

- Write a sample test asserting that true is true. Create an arrange, an act, and an assert on an object, setting its properties, initializing it in a beforeEach
- Run Unit Tests with Karma

Step 2: Isolated Tests of a Pipe

Presentation

- Isolated Unit Tests

Demonstration

- strength.pipe.spec.ts
- Write ‘Should display weak if strength is 5’ test

Code

- https://github.com/joeeames/UnitTestingAngularLikeABoss/blob/solution/src/app/strength/strength.pipe.spec.ts

Tasks

- Write tests for the other 2 levels of strength

Step 3: Isolated Tests of a Service

Demonstration

- message.service.spec.ts`
- Write ‘should have no messages to start’
- Write ‘add should add a message’

Code

- https://github.com/joeeames/UnitTestingAngularLikeABoss/blob/solution/src/app/message.service.spec.ts

Tasks

- Do problem 1
- Write a test for the clear method

Step 4: Isolated Tests of a Component with Mocking

Demonstration

- Heroes.component.isolated.spec.ts
- Write ‘should remove the selected hero from the heroes list’

Goals

- Learn component testing, mocking, Code
- https://github.com/joeeames/UnitTestingAngularLikeABoss/blob/solution/src/app/message.service.spec.ts

Tasks

- Do Problem 2, 3 AND 4

Step 5: Interaction Testing

Presentation

- Interaction Testing

Demonstration

- Heroes.component.isolated.spec.ts
- Write ‘should call deleteHero

Goals

- Learn interaction testing

Code

- https://github.com/joeeames/UnitTestingAngularLikeABoss/blob/solution/src/app/message.service.spec.ts

Tasks

- Adjust test ‘should call deleteHero’ to check the correct parameter was used
- Do Problem 5

Step 6: Integration Tests

Presentation

- Integration Tests

Demonstration

- Debugging
    - Use --sourcemaps=false in npm test
    - Open up console in karma
- Write the hero.component ‘should have the correct hero’ test

Goals

- Learn basic integration tests
- Learn to test rendered HTML

Code

- https://github.com/joeeames/UnitTestingAngularLikeABoss/blob/solution/src/app/hero/hero.component.shallow.spec.ts

Tasks

- Modify the test to check the correct parameter was passed

Step 7: Testing Rendered HTML

Presentation

- Querying the DOM 1 slide

Demonstration

- Write the hero.component ‘should render hero name in an anchor tag
- Show how to solve with BOTH nativeElement and DebugElement
- Discuss both methods of By

Goals

- Learn DOM Querying
- Learn how to call change detection

Code

- https://github.com/joeeames/UnitTestingAngularLikeABoss/blob/solution/src/app/hero/hero.component.shallow.spec.ts

Tasks

- Add a test to check that the hero ID is correct. Use both native element & debug element

Step 8: Mocking Services & Child Components

Demonstration

- Write the heroes.component.shallow ‘should set heroes correctly from service’

Goals

- Learn to mock a service
- Learn to mock a subcomponent

Code

- https://github.com/joeeames/UnitTestingAngularLikeABoss/blob/solution/src/app/heroes/heroes.component.shallow.spec.ts

Tasks

- Add a test to count the number of heroes by counting the number of li tags
- Do Problem 4

Step 9: Deep Integration Tests

Presentation

- Deep Integration Tests - Deep Component Testing & Access to the child Components

Demonstration

- Write heroes.component.deep ‘should render each hero as a hero component’, **only testing the first hero** - name property, then whole hero
- DO NOT WRITE ROUTERLINK STUB

Goals

- Learn to query by directive
- Learn Deep Testing

Code

- https://github.com/joeeames/UnitTestingAngularLikeABoss/blob/solution/src/app/heroes/heroes.component.deep.spec.ts

Tasks

- Adjust the test to test all heroes

Step 10: Test a service with an Integration Test

Presentation

- Integration Testing of Services

Demonstration

- Write heroes.service.spec ‘should call with the correct URL’

Goals

- Learn to use mock httpClient & controller

Code

- https://github.com/joeeames/UnitTestingAngularLikeABoss/blob/solution/src/app/hero.service.spec.ts

Tasks

- Write a test for searchHeroes

Step 11: Trigger an event

Demonstration

- Write ‘should call heroservice.deleteHero when the HeroComponent's delete button is clicked’

Goals

- Learn multiple ways to trigger an event

Code

- https://github.com/joeeames/UnitTestingAngularLikeABoss/blob/solution/src/app/heroes/heroes.component.deep.spec.ts

Tasks

- Write a test for adding a new hero to the list
-

Step 12: Testing Async Code

Presentation

- Asynchronicity in Unit Tests

Demonstration

- Write hero-detail.component.spec test “should updateHero when save is called”
- New save method with promise implementation
- 3 versions

Goals

- Learn to test Async code using Angular Helpers

Code

- https://github.com/joeeames/UnitTestingAngularLikeABoss/blob/solution/src/app/hero-detail/hero-detail.component.spec.ts

Tasks

- Write a test that checks that goBack is called by the save method

Summary Exercises

- Write an isolated test for hero-search.component.ts search method
    - Will require good knowledge of RXJS, Angular, and unit testing
- Write an integration test for hero-search that types in the text searchbox.
    - Check that it creates the right number of li’s

Step 13: BONUS

Cypress.IO

OR

Test a routing component

Demonstration

- Write the heroes.component.deep RouterLinkDirectiveStub
- Write ‘has the correct route for the first hero’

Goals

- Learn to test a component with a routerLink
- Learn to get a handle to a directive

Code

- https://github.com/joeeames/UnitTestingAngularLikeABoss/blob/solution/src/app/heroes/heroes.component.deep.spec.ts

Tasks

- Write a test for the hero-search component to check the URL of a hero