AngularJS

Unit Testing with Karma & Jasmine

Getting started with testing Angular

- AngularJS is built with testing in mind
- Testing is a good approach to reap the following benefits
    - keep code maintainable,
    - Keep code understandable,
    - Keep code debug-able, and
    - Keep code bug-free
- Testing can make our software more reliable, more fun, and help us sleep better at night
- Good test suites can help us find problems before they appear in production

How and when to test

- Test-Driven Development | TDD:
  - Write tests first
  - Write a test to match the functionality & API we expect out of our element
- Write-Behind Testing | WBT:
  - Write tests last
  - Write tests to confirm the functionality works as expected
- Black-Box Testing:
  - Write tests to black-box test the functionality of the overall system

Karma Test Runner

- Karma – A JavaScript test runner that fits the needs of an AngularJS developer
  - Developed by the AngularJS team

- Karma is testing framework agnostic
  - Can describe your tests with Jasmine, Mocha, Qunit
  - Can write adapter for framework of your choice

- Karma allows you to test your code on real devices
  - You can test your code on real browsers and real devices such as phones, tablets or on a headless PhantomJS instance.

Setting up Karma

- install karma for development purposes

```
$ npm install --save-dev karma
```

- Install karma command line interface globally

```
$ npm install -g karma-cli
```

- Install karma plug-ins to enable us to use Jasmine test framework and Google Chrome as target browser

```
$ npm install jasmine-core karma-jasmine
  karma-chrome-launcher --save-dev
```

Configuring Test Runner

- Create a configuration file for the karma settings

```
$ karma init karma.conf.js
```

- You will be asked several questions
- Accept the defaults to as many as you can
- **Answer NO for the RequireJS question**
- **Will fill in the source and test files section manually**
- The config file called *karma.conf.js* will be created
- Will use cofig file to run run tests from the terminal

Files section of config file

```
// list of files / patterns to load in the browser

files: [
      'node_modules/angular/angular.js',
      'node_modules/angular-mocks/angular-mocks.js’,
      './*.js',
      'tests/*Spec.js'
],
```

- Install angular-mocks to inject and mock Angular services into your unit tests

```
$ npm install angular-mocks --save-dev
```

Running unit tests

- Start test runner by issuing following command

```
$ karma start karma.conf.js
```

- Expect tests to fail (none written) & fix fixable errors
- Optimization: update the *package.json* manifest with *scripts* section to run karma

```
$ npm test
```

Scripts section of manifest

```
"scripts": {
     "test": "karma start karma.conf.js”
},
```

Testing with Jasmine

- Since karma works well with Jasmine, we'll be using the Jasmine framework as the basis for our tests.

http://jasmine.github.io/2.4/introduction.html describes the Jasmine testing framework.

- A behavior driven testing framework for testing JavaScript code
- Suites **describe** Your Tests
- Specs say what **it** must do to perform the tests

- Expectations are like assert statements
- They expect actual values to match
- You can group related specs with describe
- Can do setups before and teardowns for test suites
- Test doubles called Spies are available in Jasmine

Testing AngularJS controllers

- Create a test suite with **describe**.
  - The string parameter should include the name of the controller being tested.
  - The function parameter is the block of code that implements the suite

- Use **beforeEach** to load the module that contains the controller being tested.

- Inject the **$controller** and **$rootScope** services in a **beforeEach** block
  - That allows you to create a new $scope and the controller.
  - Attach new scope to the controller
  - We can interact with the $scope throughout the tests

Testing AngularJS controllers (2)

- Now that everything is setup, we can **spec** out tests using the **it** function.
  - String parameter is title of spec or description of what the spec is testing
  - Function parameter is the spec or test.

- Test functionality of code that we write
  - Write tests for variables can be manipulated by the user
  - Write tests for running custom actions that we implement
  - Don't write tests for simple JavaScript that we know work
  - Each test should verify a Specific Functionality or behavior
  - Might have to mock up services and other injectable dependencies

- Each test should have 1 or more expectations
  - Might be wise to follow this testing paradigm: setup → run code → assert

Examples

- Walk through process of creating and running controller tests for sample application.

Resources

- http://karma-runner.github.io/0.13/index.html
- http://jasmine.github.io/2.4/introduction.html
- http://www.ng-newsletter.com/25-days-of-angular/day-19
- http://www.bradoncode.com/blog/2015/05/19/karma-angularjs-testing/