Test-Driven Development for Angular

bit.ly/

rockncoder-2018-rwx-ng-tdd

Troy Miles

- Troy Miles
- Nearly 40 years of experience
- Programmer, speaker & author
- rockncoder@gmail.com
- @therockncoder
- lynda.com Author

Agenda

- Jasmine
- Components
- Directives
- Pipe
- Summary

What we aren't covering

- Protractor
- Selenium
- Acceptance testing
- Behavior Driven Development

My Versions

app	command	my version
git	git —version	2.15.0
node.js	node -v	v8.11.1
npm	npm —v	5.6.0
angular cli	ng -v	1.7.4

Know Your Tools

Tool	Command
Web Server	ng serve
Unit Test	ng test
End to End Test	ng e2e
Dev Build	ng build —dev
Production Build	ng build —prod

Create New Components

Component	Command
Class	ng g class my-new-class
Component	ng g component my-new-component
Directive	ng g directive my-new-directive
Enum	ng g enum my-new-enum
Guard	ng g guard my-new-guard
Interface	ng g interface my-new-interface
Module	ng g module my-module
Pipe	ng g pipe my-new-pipe
Service	ng g service my-new-service

Setup

- Multi-window setup
- IDE
- Web browser
- two terminal/command windows
- (terminals at app root)

Jasmine

- Latest version 2.5.3
- The default unit test for Angular
- Behavior-driven JavaScript

describe() - Test Suite

- describe() is a global Jasmine function
- Takes to parameters
  - name of the test suite (string)
  - implementation of the suite (function)
- Can be nested

describe()

```
describe('App: Quizzer', () => {
  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [
        AppComponent
      ],
      imports: [
        RouterTestingModule
      ]
    });
  });
});
```

it() - Test Spec

- it() is a global Jasmine function
- Takes two parameters
  - name of the spec (string)
  - implementation of the spec (function)

it()

```
it(`should have as title 'Quizzer'`, async(() => {
  let fixture = TestBed.createComponent(AppComponent);
  let app = fixture.debugElement.componentInstance;
  expect(app.title).toEqual('Quizzer');
}));
```

expect() - Expectation

- expect() is a global Jasmine function
- Jasmine's version of assert()
- Takes one parameter
  - The actual - value generated by code under test
- Is chained to a Matcher

Matcher

- Takes the output of the expect() function
- Takes one parameter
  - The expected value
- Compares the expect and actual value using the logic of the matcher

expect()

```
let app = fixture.debugElement.componentInstance;
expect(app).toBeTruthy();
expect(app).not.toBeUndefined();
expect(app.title).toEqual('Quizzer');
```

Matchers (part 1)

Matcher	Comparison
toBe()	compares using ===
toEqual()	works for literal variables and objects
toMatch()	for regular expressions
toBeDefined()	compares against 'undefined'
toBeUndefined()	also compares against ‘undefined'

Matchers (part 2)

Matcher	Comparison
toBeNull()	compares against null
toBeTruthy()	truthy boolean casting
toBeFalsy()	falsy boolean casting
toContain()	finds an item in array

Matchers (part 3)

Matcher	Comparison
toBeLessThan()	mathematical comparison
toBeGreaterThan()	mathematical comparison
toBeCloseTo()	precision math comparison
toThrow()	should throw an exception

Angular's Matchers

Matcher	Comparison
toBePromise()	the value is a promise
toBeAnInstanceOf()	an instance of an object
toHaveText()	the element has the given text
toHaveCssClass()	the element has the given CSS class
toHaveCssStyle()	the element has the given CSS styles
toImplement()	the class implements the given interface

Custom Matchers

```
var customMatchers = {
  toBeGoofy: function (util, customEqualityTesters) {
    return {
      compare: function (actual, expected) {
        if (expected === undefined) {
          expected = '';
        }
        var result = {};
        result.pass = util.equals(actual.hyuk, "gawrsh" + expected,
customEqualityTesters);
        result.message = result.pass ?
        "Expected " + actual + " not to be quite so goofy" :
        "Expected " + actual + " to be goofy, but it was not very goofy";
        return result;
      }
    };
  }
};
```

beforeEach()

- Setup function
- Called before each spec is executed
- A good place to add customer matchers

beforeEach()

```
beforeEach(function() {
    jasmine.addMatchers(customMatchers);
});
```

this

- beforeEach sets the *this* construct to any empty object

- It is passed to each it() and afterEach()

- The modified *this* doesn't flow thru from one it() to the next

afterEach()

- Teardown function

- Called after each spec is executed

Disabling suites & specs

- prepend an ‘x’
- to disable a suite change describe() to xdescribe()
- to disable a spec change it() to xit()
- They are not execute but appear in reporting

Dependency Injection is not Magic

- Dependency Injection (DI) is a core component of Angular
- It only means objects are instantiated somewhere else
- And supplied to your code when it needs them
- Angular calls DI a provider

providers

```
@NgModule({
  imports: TroyModules,
  declarations: [
    AppComponent,
    AboutComponent,
    LoginComponent,
    QuizComponent,
    PlayerComponent,
    MixedPipe,
    BackcolorDirective,
    ModelComponent
  ],
  providers: [QuizService],
  bootstrap: [AppComponent]
})
export class AppModule {
}
```

Module Initialization

Key	Value
imports	Modules (array)
declarations	Components (array)
providers	Services and other DI objects (array)
bootstrap	the launch component

Unit Tests & Providers

```
describe('Router', () => {
  beforeEach(() => {
    // acts like NgModule
    TestBed.configureTestingModule({
      imports: [
        RouterTestingModule, AppRoutingModule
      ],
      declarations: [
        AppComponent, AboutComponent, LoginComponent
      ],
      providers: [
        {provide: APP_BASE_HREF, useValue: '/'}
      ]
  });
});
```

Testing Components

Unit Test & DI

- When unit testing your app you ask Angular for just enough parts to make it work

- Angular's injectors use DI to do this

Components

- We need to create the component via createComponent
- Replace outside things like services and directives, with mocks
- Then test the components functionality

TestBed

- Angular testing module of type @NgModule
- Parameters similar to @NgModule
- Configured via its configureTestingModule method
- Configured during beforeEach so it reset before each test run

createComponent

- Creates an instance of the component under test
- Closes the TestBed to further configuration
- Returns a ComponentFixture object

compileComponents

- Asynchronously compiles external views and CSS

- Not needed if you are using Webpack since it inlines views and CSS

Testing Directives

Attribute Directives

- The challenge with attribute directives is that they don’t exist in isolation

- Resist the urge to test them with one of your app’s components

- Instead create a test component for it

DebugElement

- It is essentially an API to the component's test DOM
- It can be queried
- It can inject a service or other injectable
- It can trigger an event

Isolated Unit Testing

- If subject under test doesn't:
  - Import from Angular test library
  - Configure a module
  - Prepare DI providers
  - Call inject or async
- Then you should isolated unit test it

Isolated Unit Testing

- Means you can test it by itself

- Services and pipes are often good candidates

Testing Services

Services

- Isolated unit testing works well with services
- No need to use an injector simply new the service

Testing Pipes

Pure Functions

- Must take parameters and return a value
- Return value depends solely on parameters
- Does not cause side-effects
- Does not mutate parameters
- Are easy to unit test because of the above

Pipes

- Pipes transform displayed values within a view
- The class has one method, transform
- Usually they are written as pure functions
- Pure function are by definition, easy to test

Http Testing

- HttpClient replaces Http

- Introduced with Angular v4.3.0

Router Testing

- Test components that use the router

- Test the router

Useful Links

- https://angular.io/
- https://jasmine.github.io/index.html
- https://testing.googleblog.com/2015/04/just-say-no-to-more-end-to-end-tests.html
- https://www.ng-conf.org/mocking-dependencies-angular/
- https://kirjai.com/testing-angular-services-with-dependencies/

bit.ly/

rockncoder-2018-rwx-ng-tdd

Summary

- Unit Test help to make your code more solid
- Writing the tests first, means that you don’t have to write them later
- Writing tests first can lead to less code