Modern Web Development for Java Programmers

Unit 4. TDD for JavaScript.
Testing AngularJS components. AJAX, JSON, REST

Unit 4 Timeline

• Code Quality Tools (jshint, tslint)	15 min
• Test Driven Development	10 min
• Jasmine	10 min
• Hands-on: Using Jasmine	10 min
• Test Runners: Karma	10 min
• Hands-on: Initializing Karma project, run tests, IDE integration	20 min
• AngularJS specific testing	20 min
• Hands-on: TDD with AngularJS	20 min
• Break	10 min
• AJAX, JSON, REST, HATEOAS, CORS	40 min
• Angular Components: AngularResource, RestAngular	15 min
• Hands-on: Consuming JSON services from a fake server - apiary	20 min
• Home work: cover components with tests, consuming JSON from the server via HTTP protocol.	

Code Quality Tools

Code Quality Tools

- JavaScript is dynamic and interpreted. There is no compiler to help, for instance, like in Java
- Maintaing team-wide code style
- Apply the best practices from community to make your code better

- JSHint http://www.jshint.com/docs/
- TSLint https://github.com/palantir/tslint

Further reading

- «Selected Productivity Tools for Enterprise Developers» chapter, http://enterprisewebbook.com/ch5_tools.html#_using-grunt-to-run-jshint-checks

- http://www.jshint.com/docs/

Testing for JavaScript

Why to test ?

- Maintain code quality
- Prevent regressing and implementing new features without breaking the existing ones
- Testable systems overall have better design and understandable code

Types of testing

- Unit testing
- Integration testing
- Functional testing
- Load (a.k.a. stress) testing

We’re going to talk about
unit testing

What is TDD

- TDD meant writing tests before or inline with the actual implementation
- Mock complex dependencies
- Tests help track previously-fixed bugs
- Run the tests automatically and receive feedback from your code

Mocks vs Spies vs Stubs

- Mocks replace entire objects/interfaces to control data flow
- Spies replace/patch existing functions to intercept behavior
- Stubs hijack the return value of a function to control program flow
- AngularJS has mocks built-in. Spies and Stubs can be used with JavaScript

«Test first»

- Write the test and make it fail
- Make the test pass
- Refactor
- Repeat

RED

GREEN

REFACTOR

Jasmine

Behavior-driven framework for JavaScript

Behavior-driven development (BDD)

- used the natural language constructs to describe what you think your code should be doing
- specifications should be sentences
- help you to easy identify the failed test by simply reading this sentence in the resulting report

Walkthrough 1

Jasmine

- Folder walkthroughs/w1
- Use `bower install` to download required dependencies
- Open *test.html* file in a browser
- Examine test/spec folder and make yourself familiar with Jasmine specs syntax

Walkthrough recap

- Basic concepts (Specs, Suites)
- Matchers
- Setup and Teardown
- Jasmine HTML-runner
- Jasmine support in IntelliJ IDEA (add jasmine library support, generate suite, spec, beforeEach, afterEach)

Run the tests

with Karma

What is Karma

- Designed to run simple tests very fast
- Works with multiple browsers simultaneously
- Collects and displays total results
- Works great with AngularJS
- Can watch for changed files and execute tests automatically

debugging your tests

- `console.log()`
- `alert()`
- `dump()`
- `debugger;`

Walkthrough 2

Jasmine, Karma and IntelliJ IDEA

- Folder walkthroughs/w2
- Use *`npm install`* && *`bower install`* to download required dependencies
- Run `karma start`
- Watch how karma will start all mentioned browsers
- Configure karma in IntelliJ IDEA
- Demonstrate debugging options
- **Optional:** Connect to karma server from other browsers, e.g. IE, iOS, etc

Testing the AngularJS components

angular-mocks.js

- mocking tools to easily test AngularJS modules
- `angular.mock.module()`
- `angular.mock.inject()`

angular.mock.inject

```javascript
angular.module('walkthroughModule', [])
    .value('mode', 'app')
    .value('version', 'v0.0.4');
```

```javascript
describe('walkthrough module', function () {
    beforeEach(module('walkthroughModule'));

    it("version should be v0.0.4", inject(function (version) {
        expect(version).toBe('v0.0.4');
    }));

    it("mode should be app", inject(function (mode) {
        expect(mode).toBe('app');
    }));
});
```

Testing a directive

```
app.directive('waPanel', function () {
    return function (scope, element) {
        element.addClass('wa-panel');
    }
});
```

```
describe("testing a directive", function () {
    var element;
    beforeEach(inject(function ($compile, $rootScope) {
        element = angular.element('<div wa-panel></div>');
        element = $compile(element)($rootScope);
    }));
    it("should have a class panel", function () {
        expect(element.hasClass('wa-panel')).toBe(true);
    });
});
```

Testing a controller

```
app.controller("AppCtrl", function () {
    this.greeting = "Hi there";
});
```

```
describe("testing a controller", function () {
    var appController;

    beforeEach(inject(function ($controller) {
        appController = $controller("AppCtrl");
    }));

    it("should have greeting Hi there", function () {
        expect(appController.hamlet).toBe("To be or not to be");
    });
});
```

Testing a service

```
describe('testing a service', function () {
    it('location should be empty', inject(function ($location) {
        expect($location.path()).toBe('');
    }));
});
```

Testing the routes

```
app.config(['$routeProvider', function ($routeProvider) {
    $routeProvider
        .when('/', {
            templateUrl: 'views/main.html',
            controller: 'MainCtrl'
        })
        .when('/search', {
            templateUrl: 'views/search.html',
            controller: 'SearchCtrl'
        });
}]);
```

```
describe('testing a route', function () {
    describe('main route', function () {
        it('should use main.html view ', inject(function ($route) {
            expect($route.routes['/'].templateUrl).toEqual('views/main.html');
        }));
        it('should handled by MainCtrl', inject(function ($route) {
            expect($route.routes['/'].controller).toEqual('MainCtrl');
        }));
    });
    describe('search route', function () {
        it('should use search.html view', inject(function ($route) {
            expect($route.routes['/search'].templateUrl).toEqual('views/search.html');
        }));
        it('should handled by SearchCtrl', inject(function ($route) {
            expect($route.routes['/search'].controller).toEqual('SearchCtrl');
        }));
    });
});
```

Walkthrough 3

Testing an AngularJS components

- Folder walkthroughs/w3
- Use *`npm install && bower install`* to download required dependencies
- Run karma test with command line or via IntelliJ IDEA
- Examine test/spec folder and make yourself familiar with AngularJS specifics in testing

Additional resources

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

Additional resources

- «Test-Driven Development with JavaScript» chapter http://enterprisewebbook.com/ch7_testdriven_js.html
- «Growing Object-Oriented Software, Guided by Tests» http://www.amazon.com/Growing-Object-Oriented-Software-Guided-Tests/dp/0321503627
- Test-Driven JavaScript Development http://www.amazon.com/Test-Driven-JavaScript-Development-Developers-Library-ebook/dp/B004519O02
- «Testable JavaScript» http://www.amazon.com/Testable-JavaScript-Mark-Ethan-Trostler-ebook/dp/B00B1WLE92

AJAX. JSON. REST.

AJAX

- Asynchronous
- JavaScript
- and
- ~~XML, JSON, text…~~
- …whatever!

XMLHttpRequest

```
var xhr = new XMLHttpRequest();

xhr.open("GET", “http://webauctionv1.apiary-mock.com/product/featured");

xhr.onreadystatechange = function () {
    if (this.readyState == 4) {
        alert('Status: ' + this.status + '\nHeaders: ' +
JSON.stringify(this.getAllResponseHeaders()) + '\nBody: ' + this.responseText);
    }
};

xhr.send(null);
```

REST

- REpresentational State Transfer
- Addressable resources
- Representation-oriented
- Communicate statelessly
- “Hypermedia As The Engine Of Application State (HATEOAS)”

Using fake server

http://webauctionv1.apiary-mock.com/
http://docs.webauctionv1.apiary.io/

Walkthrough 4

Consuming JSON services with IntelliJ IDEA

Additional reading

- http://enterprisewebbook.com/ch2_ajax_json.html
- http://www.ng-newsletter.com/posts/restangular.html
- http://docs.angularjs.org/api/ngResource/service/$resource

Homework 4

- Implement the tests for application developed in Homework 3
- Test the application components using AngularJS support for Jasmine
- Refactor the application to use a fake server