ANGULARJS (level 200)

HTML enhanced for web apps!

Vihang Shah

Associate Project Manager @ Tatvasoft

http://vihang-shah.blogspot.com/

https://www.facebook.com/vihangshah.it

ANGULAR Basics – Revision / Future

- Revision of last training:
  - Binding
  - Controllers
  - Services
  - Expressions
  - Forms (Validation)
  - $scope vs. scope
  - What is MEAN?
- Angular 1 (v 1.5.5) vs. Angular 2beta (v 2.0.0-rc.1 (2016-05-03))
  - Cross Platform
    - Progressive Web Apps
    - Native
    - Desktop
  - Speed & Performance
    - Code generation
    - Universal
    - Code Splitting
  - Productivity
    - Universal
    - Angular CLI
    - IDEs
  - More...

Routing

It Is used for deep-linking URLs to controllers and views (HTML partials). It watches $location.url() and tries to map the path to an existing route definition.

```
$routeProvider.when('/Book', {
    template: 'examples/book.html',
    controller: BookCntl
});
$routeProvider.when('/Book/chapter01', {
    template: 'examples/chapter01.html',
    controller: ChapterCntl
});
$routeProvider.when("/:message", {
    templateUrl: "app.html",
    controller: "AppCtrl"
});
```

Dependency Injection

Dependency Injection (DI) is a software design pattern that deals with how components get hold of their dependencies.

The Angular injector subsystem is in charge of creating components, resolving their dependencies, and providing them to other components as requested.

Promises

- Promises in AngularJS are provided by the built-in $q service.
- They provide a way to execute asynchronous functions in series by registering them with a promise object.

$http and Server Interaction

- The $http service is how AngularJS makes rest API calls. It's capable of making the common 'GET', 'POST', 'PUT', and 'DELETE' API calls as well as the less common 'PATCH' and 'HEAD' calls. It can also make jsonp calls for cross-origin requests.
- The Response Object
  - The response object passed back via the promise contains several properties that describe the request:
    - **config**: the initial config object used to make the request
    - **data**: the data returned from the server. If the response type is set to 'application/json', Angular will automatically parse the received data and make it available as an object via the **data** attribute associated on the response object.
    - **headers**: this property contains further functions for retrieving header information associated with the request
    - **status**: the HTTP status code of the request
    - **statusText**: accompanying text of the response (e.g. "OK", "Bad Request", etc.)
- success() and error() Methods
  - The promises returned by $http have two additional non-standard methods attached to them: **success**() and **error**().
  - These methods behave just like the then() and catch() methods on standard promises accept rather than passing a standard response object to the callback function, they instead pass the response properties as function parameters starting with data.

Filters

Angular filters format data for display to the user.

Filters are a simple but powerful tool in Angular, used primarily to format expressions in bindings in views and templates.

```
<body ng-app="app" ng-controller="TestCtrl as test">
    <input type="text" ng-model="search" />

    <p ng-repeat="person in test.people | filter: search">
        {{ person.name }}
    </p>
</body>
```

Other Examples:

```
{{ expression [| filter_name[:parameter_value] ... ] }}
{{ uppercase_expression | uppercase }}
{{ expression | filter1 | filter2 }}
```

We can also create custom filters

Directives - Basic Directive Functionality

- Directives are certainly one of the most important facets of the framework
- With directives, you can create custom, reusable components for your application that make developing interactive UI widgets drastically easier.
- The directives can be placed in element names, attributes, class names, as well as comments. Directives are a way to teach HTML new tricks.
- A directive is just a function which executes when the compiler encounters it in the DOM.

```
<input ng-model='name'>
```

- Custom Defined Directives

```
angular.module('functionalities', [])
    .directive("entering", function () {
        return function (scope, element) {
            element.bind("mouseenter", function () {
                console.log("Mouse has entered the div");
            })
        }
    })

    .directive("leaving", function () {
        return function (scope, element) {
            element.bind("mouseleave", function () {
                console.log("Mouse has left the div");
            })
        }
    });
```

```
<div entering leaving>Hover over me for fun times</div>
```

Directives - Directive Communication

```
var app = angular.module("app", []);

app.directive("country", function () {
    return {
        restrict: "E",
        controller: function () {
            this.makeAnnouncement = function (message) {
                console.log("Country says: " + message);
            };
        }
    };
});
app.directive("state", function () {
    return {
        restrict: "E",
        controller: function () {
            this.makeLaw = function (law) {
                console.log("Law: " + law);
            };
        }
    };
});
app.directive("city", function () {
    return {
        restrict: "E",
        require: ["^country", "^state"],
        link: function (scope, element, attrs, ctrls) {
            ctrls[0].makeAnnouncement("This city rocks");
            ctrls[1].makeLaw("Jump higher");
        }
    };
});
```

Animations (ngAnimate)

- Include angular-animate.js in your HTML

```
<script src="path/to/angular.js"></script>
<script src="path/to/angular-animate.js"></script>
```

- Load the module in your application

```
angular.module('app', ['ngAnimate']);
```

- Then, just put.

```
<div ng-if="bool" class="fade">
    Fade me in out
</div>
<button ng-click="bool=true">Fade In!</button>
<button ng-click="bool=false">Fade Out!</button>
```

Animations (ngAnimate) continue…

The following directives are "animation aware":

Directive	Supported Animations
ngRepeat	enter, leave and move
ngView	enter and leave
ngInclude	enter and leave
ngSwitch	enter and leave
ngIf	enter and leave
ngClass	add and remove (the CSS class(es) present)
ngShow & ngHide	add and remove (the ng-hide class value)
form & ngModel	add and remove (dirty, pristine, valid, invalid & all other validations)
ngMessages	add and remove (ng-active & ng-inactive)
ngMessage	enter and leave

Templates

- In AngularJS, there are multiple ways for you to specify templates in your application.

- Using Templates as a String

```
angular.module('app', ['ngRoute'])
    .config(function ($routeProvider) {
        $routeProvider.when('/', {
            controller: 'TestCtrl as test',
            template: 'Hello {{ test.user.name }}!'
        })
        .otherwise('/');
    })
```

- Using a Remote Template

```
angular.module('app', ['ngRoute'])
    .config(function ($routeProvider) {
        $routeProvider.when('/', {
            controller: 'TestCtrl as test',
            templateUrl: 'test.html'
        })
    .otherwise('/');
    })
```

Templates continues…

- Using Inline Templates

```
<body ng-app="app">
        <div ng-view></div>
        <script type="text/ng-template" id="test.html">
            Hello {{ test.user.name }}!
        </script>
</body>
```

- Populating $templateCache Directly
  - $templateCache is simply a key-value store where you can get and put templates. Angular will automatically check $templateCache whenever you use templateUrl in your application or when you specify a template src with ng-include before attempting to retrieve your template from a remote location.

```
function TestCtrl($templateCache) {
            this.user = { name: 'Blake' };
            console.log($templateCache.get('test.html'));
        }
        angular.module('app', ['ngRoute'])
        .config(function ($routeProvider) {
            $routeProvider.when('/', {
                controller: 'TestCtrl as test',
                templateUrl: 'test.html'
            })
            .otherwise('/');
        })
        .controller('TestCtrl', TestCtrl);

        angular.module('app').run(function ($templateCache) {
            $templateCache.put('test.html', 'Hello {{ test.user.name }}!');
        });
```

Unit testing

- **Yeoman** has the scaffolding tool so that generates an application skeleton to start out with, complete with things like Bootstrap or a pre-configured testing setup. The scaffold of the application is different in many ways to the angular-seed scaffold, but it is important to note that they both use Karma and Jasmine in the same ways.
- **angular-seed** is a ready-to-eat AngularJS scaffold available on their github with directory structure and testing amenities pre-prepared. Testing this application is accomplished by running a standalone Karma test server.
- **Grunt** is the testing tool used in Yeoman, but it is used as a wrapper for Karma.
- **Karma** is the actual test runner that is used to test AngularJS. It can be used standalone from Grunt. Karma circumvents testing inconsistencies across browsers, which would happen with things like PhantomJS, by actually launching a browser and running the tests in it.
- **Jasmine** is the testing framework which is used for unit tests by default in Karma. Angular E2E tests with Karma don't and can't use the Jasmine adapter, although E2E tests use very similar syntax with the Angular Scenario Runner. This blog post does a fine job of going through how to actually author some Jasmine tests, and gives some excellent examples.

- Step by step guide: Full-Spectrum Testing with AngularJS and Karma

Coding standards

- Angular styleguide by John Papa 
  - https://github.com/johnpapa/angular-styleguide

- Angular styleguide by Todd Motto 
  - https://github.com/toddmotto/angular-styleguide

Resources

Documentation

- AngularJS API
- Things I Wish I Were Told About Angular.js
- Angular io - 2.0 features
- Full-Spectrum Testing with AngularJS and Karma

Videos

- Introduction to Angular JS
- Angular JS end-to-end web app tutorial Part I
- Angular JS ng init directive

Thank you

Vihang Shah

Associate Project Manager @ Tatvasoft

http://vihang-shah.blogspot.com/

https://www.facebook.com/vihangshah.it