AngularJS $scope, $rootScope

Review Yesterday

Questions about MVC, data binding?

Today

scope, $scope, $rootScope, $scope-less binding

JavaScript scope

Scope in JavaScript defines where certain variables, functions, and objects are available in the code. To make JavaScript variables, functions and objects available to our HTML page, they need to be tied to the global scope. Here are global and namespaced examples using standard JavaScript:

```
<button onclick="callGlobalFunction()">
   Click Me</button>
<button onclick="moduleName.callModuleFunction()">
   Click Me</button>
```

Angular $scope

$scope in Angular is an object that Angular creates for many DOM elements in the page. The $scope is used to store the model (data and functions) pertaining to that DOM element. A typical Angular page has many $scope objects, one for each of the elements that requested one.

$rootScope

Tommy Fisher describes $scope and $rootScope best:

> When your application starts, AngularJS creates the initial scope, which it calls `$rootScope`. It then "compiles" the document, starting at the root element. As it traverses the DOM, it encounters and processes markers that it calls directives, and some of these (such as `ng-controller`) request new scopes. After compilation is done, AngularJS will have created a scope tree that mirrors the DOM tree - the `$rootScope` is bound to the root element and child scopes are bound as the DOM nodes that request them are discovered. Separate scopes are incredibly useful. As seen in the example [below], they allow for different parts of the view to cleanly separate their part of the underlying model. (from carbonfive.com)

NOTE: There is always only one "global" $rootScope per Angular app.

Here is a sample DOM (html file) that has 3 $scopes: one for each controller and the $rootScope.

```
<div ng-controller="GreetController">
   Hi {{name}}!
   <div ng-controller="FarewellController">
      Bye {{name}}!
   </div>
</div>
Who is {{name}}?

angular.module('scopeExample', [])
.controller('GreetController', GreetController)
.controller('FarewellController', FarewellController);

GreetController.$inject = ['$scope', '$rootScope'];

function GreetController($scope, $rootScope) {
   $scope.name = 'Wesley';
   $rootScope.name = 'Fezzini';
}

FarewellController.$inject['$scope'];

function FarewellController($scope) {
   $scope.name = 'Buttercup';
}
```

$scope-less binding

Store your controller model (data and functions) in the controller object instead of in $scope. Then use "Controller as X" in the html to individually reference multiple controllers (where X is a variable name; see below). Angular still creates $scope objects for each controller, but we're just not using them to bind the html view and controller model. This is similar to namespaced modules in plain JavaScript, except these are "scoped" to their DIV.

```
<div ng-controller="GreetController as gc">
   Hi {{gc.name}}!
   <div ng-controller="FarewellController as fc">
      Bye {{fc.name}}!
   </div>
   Who is {{name}}?
</div>
```

```
angular.module('namedScopeExample', [])
.controller('GreetController', GreetController)
.controller('FarewellController', FarewellController);

GreetController.$inject = ['$rootScope'];

function GreetController($rootScope) {
   var gc = this;
   gc.name = 'Wesley';
   $rootScope.name = 'Fezzini';
}

function FarewellController() {
   var fc = this;
   fc.name = 'Buttercup';
}
```

Review

scope, $scope, $rootScope, $scope-less binding

Exercise

Resources

http://blog.carbonfive.com/2014/02/11/angularjs-scopes-an-introduction/

https://code.angularjs.org/1.4.7/docs/guide/scope

https://thinkster.io/a-better-way-to-learn-angularjs/controllers

Tomorrow

Using Angular's built-in directives.