Front-End Development with AngularJS

BEN LAWRENCE

Background

- Former High School Math Teacher
- Towson COSC Grad Student
- Student of Dr. Kaza's in Fall 2015 (4 months ago).
- Software Developer at RDA Corporation
  - Focus on front-end web development.

RDA

- Software Development Consulting Firm
- Good opportunity for people who like to code and enjoy working in a team environment
- Technologies used are as follows:
  - ASP.net
  - AngularJS
  - SiteCore
  - SharePoint
  - Microsoft Office (Apparently this can involve programming)
  - not an exhaustive list
- Any further questions about job opportunities specific to RDA, my email is benjaminrlawrence@gmail.com

Front-End Development

- Two big sides to front-end development.
- Style
  - CSS
  - Bootstrap
  - Responsive Design
- Function
  - Javascript
  - Libraries: jQuery, jQuery UI, moment.js, chart.js, perfect-scrollbar.js
  - Angular

jQuery

- Simple version
- Different browsers have slightly different flavors of javascript.
- Developers need to check the version of the browser to have different versions of the code run.
- Queries on the DOM are also verbose
  - document.getElementById('myUniqueElement')
- jQuery takes care of cross-browser issues.
- jQuery is much less verbose
  - Uses css selectors: $('#myUniqueElement')
- All functionality goes into associated .js file. Html is content, css is style, and js is functionality.
  - Separation of Concerns

AngularJS

- Angular abandons this approach
- Built-in directives act as attributes in html template.
- Functionality can become apparent by looking at the html.
- This cuts down on need for ID’s and Classes.
- I only use classes for styling and Id’s for inputs (to associate labels with those inputs)

Angular Websites

- The Weather Channel (weather.com)
- Lyft
- MSNBC
- Forbes
- Good Old Games (GOG.com)
- More at https://www.madewithangular.com/#/
  - Example of default Angular URL.

Comparison of the two

jQuery

```
<div id="myClickableDiv">

    Click Here!

</div>

_____________________

$('#myClickableDiv').on('click', function()
{

    //Do Stuff

});
```

AngularJS

```
<div ng-click="myFunc()">

    Click Here!

</div>

_____________________

$scope.myFunc = function() {

    //Do Stuff

}
```

Further Information

- Try to develop with only one or the other.
  - http://stackoverflow.com/questions/14994391/thinking-in-angularjs-if-i-have-a-jquery-background
- Previous Angular example is problematic
  - $scope
  - https://github.com/johnpapa/angular-styleguide
- Angular 2 gets rid of $scope completely, and it looks very different from Angular 1.
  - Typescript examples
  - Component-based

Thoughts on the Field

- DISCLAIMER: Limited Perspective
- Most of this is hearsay
- Advanced Web Dev focuses on a full-stack framework (Ruby on Rails)
- Absolutely necessary to be familiar with full-stack.
- In larger environments, there is high demand for people who are highly-skilled at one thing
- Current project: Client has a pre-built backend. I'm building a front-end that makes API calls to this backend.
- GET data. Display Data. POST data. Make it all pretty.
- This can be hard, because the database was built without any input from the frontend developers.

Further Thoughts

- Content Management Systems development is big
  - Sitecore, Kentico
- People don’t want to have to go to a developer to do things like update an image or change some text on the page.
- Also big for market analytics
- For most development, use a VM.
  - Front-end only is the exception.

And More Thoughts

- Iterative Development (i.e., agile) is really good. It is impossible to ensure that requirements written prior to development will still be the best course of actions.
- However, a project is either Agile, or it is not.
  - Make sure you understand as you are working on it.
- Do not shortchange the style aspect. For many clients, it is what they best understand in the project.
- Use Bootstrap for all formatting. It will save many headaches.
- Don’t use Boostrap so much that your website is obviously a Bootstrap website.

Demo

cchsdevmember.azurewebsites.net/#/

Angular Concepts

- Two-way binding
  - Will be apparent in sample assignment
- Single-page application
  - All pages load from within one set of <body> tags.
- Type of MVC, where all logic is client-side
  - Persistent storage must still be handled by the server.
- Dependency Injection
  - Built-in directives must be declared at the beginning of any controller/service.

Modules

- Collection of all controllers, services, and any other bits of code.
- This can be a self-contained app.
- It can also be a dependency that will injected into another app.

Controllers

- Similar to the controllers in Rails
- Handles all logic prior to presenting data in view
- Controllers dictate a scope
  - Size of scope depends on amount of functionality involved.

Services

- Services, Factories, and Providers are all similar
- Useful for functionality that will repeat many times throughout a module.
- I use it for API calls and getting/setting data across the app.

Factory vs Service vs Provider

Credit: http://tylermcginnis.com/angularjs-factory-vs-service-vs-provider/

```
app.controller('myFactoryCtrl', function($scope, myFactory){
  $scope.artist = myFactory.getArtist();
});

app.factory('myFactory', function(){
  var _artist = '';
  var service = {};

  service.getArtist = function(){
    return _artist;
  }

  return service;
});
```

```
app.controller('myServiceCtrl', function($scope, myService){
  $scope.artist = myService.getArtist();
});

app.service('myService', function(){
  var _artist = 'Nelly';
  this.getArtist = function(){
    return _artist;
  }
});
```

```
app.controller('myProviderCtrl', function($scope, myProvider){
  $scope.artist = myProvider.getArtist();
  $scope.data.thingFromConfig = myProvider.thingOnConfig;
});

app.provider('myProvider', function(){
  //Only line 45-46  are available in app.config().
  this._artist = '';
  this.thingFromConfig = '';

  //Only the properties on the object returned from $get are available in the controller.
  this.$get = function(){
    var that = this;
    return {
      getArtist: function(){
        return that._artist;
      },
      thingOnConfig: that.thingFromConfig
    }
  }
});

app.config(function(myProviderProvider){
  myProviderProvider.thingFromConfig = 'This was set in config()';
});
```

Directives

- Directives are re-usable functionalities.
  - They often have “ng” as a prefix.
  - ngApp, ngController, ngRepeat
  - No camel case in html, so use ng-app, ng-controller, ng-repeat instead.
- Custom directives can be made as well.
  - Useful for bits of code that will be reused.
  - Directives can take the form of a new element, attribute or class.
  - Comments can be made as well, but they are ill-advised.

Angular Lab

- https://github.com/benjaminrlawrence/AngularCrud

- Download the ZIP file from GitHub. Task is outlined in Word Doc.

Useful Directives for Lab

- ngApp – Tells angular that this tag (body or html usually) will be governed by the rules of Angular.
- ngController – Defines a $scope for a particular controller. Controllers can be nested. That is when the $scope gets hopelessly confusing.
- ngBind – binds a tag to a piece of data. This uses 2-way binding, so it updates in real-time.
  - If that scope variable is called myVar, there are 2 ways to do this:
    {{myVar}}
    or

- ngCloak – Allegedly prevents angular expressions from loading until all values are resolved. Doesn't seem to work as nicely as I'd like it to.

More Useful Directives for Lab

- ngIf – Determines if a bit of html will be instantiated.
  - Different from ngShow and ngHide, since that html is there, but with a display value of none.
  - ngIf has better performance value, but sometimes for things to work right, your html needs to be present in the DOM.
- ngInclude – Inserts template from outside the page
  - Helps make html code more readable
  - Creates a new scope, so there can be coding difficulties.
- ngModel – put on an input for that value to be recorded in that variable.
- ngSubmit – Used to determine the action taken upon submission of a form.
- ngClick – Determines action taken upon a click event.
- ngRepeat – Iterates html content over an array. Uses syntax ng-repeat="cow in cows". Now, each element of cows can be accessed with the term cow.

Useful Directives that are not in the Lab

- ngClass – Applies a class to an element given a set of conditions
  - Useful for conditional styling.
- ngSrc – allows a variable (i.e., url) to be loaded into the src attribute of an element.
  - <img ng-src="{{url}}">
- Angular has directives that interact with inputs and forms.
  - e.g., check if something is $pristine or $dirty
  - Also built-in checks using $error and $valid.