Angular 2 Learning Resources

Scott Davis

Version 0.1.0, 2016-03-29

Here are some additional learning resources to help you out with the underlying core tech used in this project.

Angular 2

http://meaningfuljs.com/
An easy command-line wizard written with the Yeoman library to scaffold out new Angular 2 projects. If you want to focus on writing applications and not worry about bootstrapping new projects with build scripts, directory structures, test harnesses, etc., then MeaningfulJS is what you're looking for. NOTE: Be sure to install the generator for Angular2 — there is a legacy Angular 1 generator as well.

https://angular.io/docs/ts/latest/quickstart.html
The Angular2 QuickStart / Case Study are great resources to learn the new framework.

https://github.com/ThirstyHead/angulario-examples-es6
We'll be using the latest version of JavaScript (ECMAScript 2015 6th Edition, or ES6) instead of TypeScript. For the Angular2 QuickStart / Case Study example implemented in ES6 instead of TypeScript, see the above GitHub repo.

https://www.ng-book.com/2/
"ng-book 2" is the most complete 3rd party book on Angular 2 at the moment. It's not without its flaws (I don't like their source code organization or naming conventions, and they are using TypeScript), but it's the best resource for more advanced concepts for now.

Since their example code isn't public (e.g. on GitHub), I don't feel comfortable hosting a public repo with ES6 examples. But if we wanted to create a private repo internally, we could share ES6 ports of all of their TypeScript examples.

https://www.youtube.com/playlist?list=PLOETEcp3DkCoNnlhE-7fovYvqwVPrRiY7
ng-conf is a non-Google event, but is considered one of the better Angular-related conferences in the US. They've posted all of their talks from 2015 on YouTube for free.

https://www.youtube.com/watch?v=-dMBcqwvYA0&list=PLOETEcp3DkCoNnlhE-7fovYvqwVPrRiY7&index=21
An especially good 45 minute talk from ng-conf 2015 is "ng-conf 2015 Keynote 2 - Misko Hevery and Rado Kirov". Misko Hevery is the Angular Project Lead, and he does a really good job explaining the changes the team made between Angular 1 and 2 — the thought process they went through in making syntax changes, etc.

ECMAScript 6

ES6 is the latest version of JavaScript. It is (nearly) fully enabled in NodeJS, and is (nearly) fully enabled in all modern browsers as well. It is the language we'll be using in this project.

https://github.com/lukehoban/es6features
A nice learning resource for new ES6 features with code examples.

http://es6-features.org/
A terse, code-only tour of ES6 features.

https://github.com/ThirstyHead/es6tests
Working examples of ES6 code, implemented as unit tests.

http://shop.oreilly.com/product/0636920050742.do
"What's New in JavaScript: A Deep Dive into ECMAScript 2015, 6th Edition" A 2 hour video tutorial by Scott Davis giving you a guided tour of new ES6 features. The first 20 minutes are free.

https://kangax.github.io/compat-table/es6/
A list of all ES6 / ES7 features currently implemented in browers, NodeJS, etc. Pay special attention to the Compilers / Polyfills section at the start of the table. Polyfill libraries "fill in" the missing ES6 features in browsers that haven't fully implemented them yet. If a feature is "green" based on a polyfill, you should feel comfortable using that feature today.

https://babeljs.io/
Babel is the ES6/ES7 polyfill library that we'll be using on this project.

Web Components

On the https://angular.io homepage, they say, "In Angular you display data by defining components." In other words, Angular2 is All Components, All The Time…

If you created custom Directives in Angular 1, Components in Angular 2 are dramatically simpler. And the new name "Components" wasn't chosen lightly — Angular 2 Components are implemented using Web Components (e.g. core browser APIs) under the covers.

"Web Components" is an umbrella term for 4 related APIs that will be baked into modern browsers as core technology: - Custom Elements - HTML Imports - Templates - Shadow DOM

These APIs can be used independently, but when used in a coordinated manner, you have the tools you need to add your own custom HTML elements to your application. In other words, instead of , you can create <person name="Suzy Q">.

Learning about Web Components and Polymer (a thin "helper" library over the core Web Components APIs) will help get you in the proper mindset for writing Angular 2 Components.

http://webcomponents.org/
The tagline says it all: "a place to discuss and evolve web component best-practices".

http://jonrimmer.github.io/are-we-componentized-yet/
The tagline says it all: "Tracking the progress of Web Components through standardisation, polyfillification, and implementation."

http://shop.oreilly.com/product/0636920050759.do
"Using Web Components: A New Direction for Modern Web Development" A 1 hour video tutorial by Scott Davis giving you a guided tour of writing your own Web Components. The first 20 minutes are free.

https://www.youtube.com/watch?v=AbunztfV5vU&list=PLOETEcp3DkCoNnlhE-7fovYvqwVPrRiY7&index=6
A good 20 minute ng-conf 2015 talk is "Creating Container Components with Web Components and Angular Kara Erickson & Rachael L Moore". This talk helps get you into the Component mindset you'll need to have to be successful with Angular 2.

https://www.polymer-project.org/1.0/
The tagline says it all: "Polymer 1.0 has been rebuilt from the ground up for speed and efficiency. The new, leaner core library makes it easier than ever to make fast, beautiful, and interoperable web components." It is a great resource for finding pre-built, ready-to-use web components.

https://elements.polymer-project.org/
The catalog of ready-to-use Polymer web components, including source code examples and live demos. Pay special attention to the "Paper" elements — these are web components that implement the Google Material Design look and feel.

https://technology.amis.nl/2016/02/10/how-to-use-polymer-webcomponents-in-angular2/
Angular 2 is "component agnostic" — you can fully intermingle Angular 2 components, Polymer components, and core Web Components all in the same app. This tutorial shows you how to use a Polymer component in an Angular 2 app.

Material Design

Angular 2 is completely agnostic when it comes to choosing a RWD (Responsive Web Design) CSS library. Twitter Bootstrap is a popular choice, but we've chosen Google Material Design for this project. MD transcends the browser — there are MD libraries for Android development, .NET development, etc. that allow you to provide your end users a consistent Look and Feel across languages and platforms.

http://www.getmdl.io/index.html
If you want a Bootstrap-like developer experience (i.e. divs and spans that use multiple CSS classes for styling), check out MDL (Material Design Lite).

https://elements.polymer-project.org/browse?package=paper-elements
Polymer offers a set of ready-to-use Material Design web components called "Paper Elements". While MDL and Polymer don't offer full feature parity, I think that Angular 2 developers will prefer the Paper Elements solution in the long run since it reinforces the Component-oriented development paradigm. NOTE: There is nothing stopping you from utilizing both MDL and Polymer in the same app.

RxJS / Observables

So, now that Promises are finally native to ES6 (and therefore available natively in all modern browsers), the Angular 2 team is already bored with them and has moved on… #sigh

In all seriousness, Reactive Programming is gaining mindshare in the development community, especially among functional programming affectionados. Reactive is a framework that has implementations in multiple languages — C#, Java, etc. The JavaScript implementation that Angular

2 requires is RxJS.

If you are writing Angular 2 Services, any AJAX call you make will be wrapped in an RxJS Observable. Think of an Observable as a Promise on steroids — you still chain calls together, but Observables includes missing Promise features like retries and accepting multiple responses from a Stream. (Promises only accept a single response.)

https://github.com/Reactive-Extensions/RxJS
The goto resource for Reactive Extensions for JavaScript (RxJS). This site includes download / installation instructions plus extensive documentation.

https://gist.github.com/staltz/868e7e9bc2a7b8c1f754
"The introduction to Reactive Programming you've been missing" Plainspoken examples of Reactive Programming. The video tutorial of this paper is quite nice as well (referenced in the introduction).

https://www.youtube.com/watch?v=5uxSu-F5Kj0
A **great** 60 minute HTML5DevConf presentation by Jafar Husain called "Asynchronous JavaScript at Netflix". I read several tutorials on RxJs prior to watching this, but this presentation is what really best helped me consolidate and integrate the information. Lots of really nice, clear code examples.

http://rxmarbles.com/
A fun visualization tool for all of the various RxJS functions.

https://angular.io/docs/ts/latest/guide/server-communication.html
Once you have a better understanding of Observables, this chapter from the Angular 2 documentation explains how to use the HTTP Angular 2 library, complete with numerous code examples that utilize Observables.