Angular

* TypeScript Framework for developing front end applications/websites
* Google created Angular
* AngularJS is the original Angular
  + Written in JS
  + Not supported by Google
  + Rather than make major updates to AngularJS Google just released Angular 2.0 which is now called Angular
* SPA (Single page Application)
  + The entire website is a single html page (usually the index .html)
  + JS will dynamically swap in and out HTML to create elements and emulate the idea of moving between pages
* Web-Component based framework
  + Create custom html tags
    - <home-page> </home-page>
    - <reimbursement-form></reimbursement-form>

Angular Folder structure anatomy cheat sheet

* e2e (end to end) where your testing folders will go
  + We will not be using this right now
* Node Modules
  + JS dependencies for you Angular Application
  + No need to touch them
* Src
  + You source code
  + 99% of your time will be in here
    - Index.html
      * That one page everyone will see
    - App folder
      * Hold all of your compoenents
* Package.json
  + Node version of the pom.xml
  + Tracks things like the name of the project
  + Dependencies for the project
* Tsconfig.json
  + Contains configuration details for typescript

Angular Hierarchy

* Every Angular application has an index.html
* Angular is modular
  + Modules are large bundles of TS/HTML/CSS
* The app.module.ts is the main module
  + It is what will reference your components

Directives

* Directives are custom HTML
  + Components
    - Custom HTML tags
  + Structural Directives
    - ngIf and ngFor
  + Attribute directives
    - [(ngModel)], ngStyle

Data Binding

* Process by which values in the TS are bound or connected to different elements in the HTML
* 1-way data binding
  + Interpolation {{}} TS => HTML
  + Event binding (click) HTML => TS
  + Attribute binding [src] = “tsVariable” TS =>HTML
* 2-way data binding
  + Where you html and TS share the value simultaneously
  + [(ngModel)]

Component

* Components are logical chunks of HTML and TS
* All components are composed of three parts
  + HTML view (visual part of a component)
  + TS class (Logical part of a component)
  + Component decorator (meta information for a component)
    - Decorator are any of the @ symbols
    - Decorators fancy functions that you can use to craft custom functions\*

Pipes

* Pipes are a way to transform you data
* Used very often for modifying your information before being displayed
  + Think formatting date/currency/weight

Services

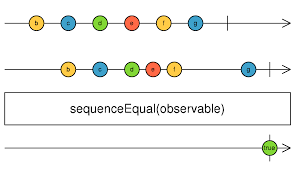
* Classes that do not have any visual aspect to them
* They provide helpful methods that you might want throughout the application
* All services are shared
  + Every service is a singleton
  + Services can store information
* Services are injected into components when components are created
  + Dependency injection (when a class is created with the objects it needs when it is constructed)

Steps to make an http call

1. Import the HttpClientModule in the app.module
2. Import HttpClient into your service
3. Inject via the constructor HttpClient into the service
4. http.get().toPromise() // turns observable into promise

Observable vs Promises

* Observables are more flexible and powerful
* Observables can do everything a promise but vice versa
  + Observables can be used for a stream of data
  + Promises are single use



* Promise is like sending a messenger and getting back an empty envelope that will eventually contain something
* Observable will set up shop at the endpoint you request and handle every response

WebPack

* HTML TS bundler
* Ultimately what takes your ton of files and compiles them together for a comprehensive single page application

Routing in Angular

* Routing modules that stores your routes
* Each route object is just a string and a component
* When someone searches for that path in the browser Angular will swap out the router outlet with the component that that path links to