Node.js

- When working Angular, we need to have Node.js installed
- Node.js is often used for server-side applications
  - You would build out a backend and run it using Node
- Node.js: We can analogously think of Node as a “JRE” (Java, which has runtime libraries and a JVM)
  - Node runs JavaScript from a non-browser environment through the V8 engine (which is also the same engine that the Chrome browser)
  - Node provides various libraries that allow you to do things beyond just the ECMAScript specification for JavaScript

ECMAScript

- This is just the standard for JavaScript
- But if we want JavaScript to do more things, such as
  - Process HTTP requests/responses
  - Read files from the system
  - Connect to a database
- We would need to extend beyond just the ECMAScript specification
  - In fact, this is what our browser already does
  - And we have actually encountered this during P1
    - DOM Manipulation: IS NOT a part of the ECMAScript specification
      - Our browser is actually extending additional functionality on top of “JavaScript”

How do functionalities get extended on top of Node.js or the Browser?

- In the case of Node.js, we are running the V8 engine
- In the case of Chrome at least (which is the most popular browser), we are ALSO using V8 (because V8 was designed originally for Chrome by Google)
    - The V8 engine was written in C++
- So, to extend functionality on top of ECMAScript, we can actually write C++ code, and then link that with our JS (to put it simply)

Why do we use Node.js with Angular?

- Node.js provides us an entire ecosystem composed of
  - Npm: node package manager, used to manage our project and various dependencies
  - Webpack: Angular uses this in the background to bundle together all of our css, html, typescript, etc. into a compact group of .js files and an index.html file (which will then serve as our SPA (single page application)
  - Angular CLI: a command-line interface that allows us to easily create a base Angular project, or add components, modules, and services to our current Angular project

Demos to reference

- Webpack Demo
- TypeScript demo
- Angular Demo

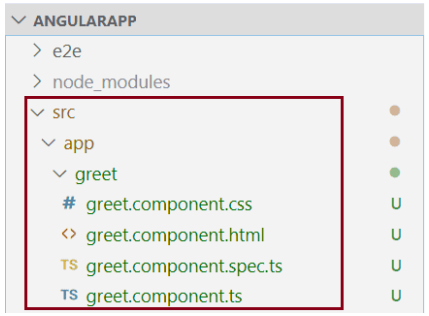
Angular Versions

- The very first “version” of Angular was called AngularJS. AngularJS is completely different from what we now refer to as Angular (without the JS)
  - AngularJS
  - Angular 2 through 11 (versions 2 through 11 are much more similar to each other than AngularJS)
    - You can think of 2, 3, 4, 5, …, 11 as different iterations of the same Angular
    - Each version here is just adding new features
- So, oftentimes you will hear people just refer to Angular 2 v. AngularJS
- AngularJS is older, and completely architecturally different

Angular Components

- Angular apps are made up of different components
- Each component is supposed to do “one thing” in the application that is visible on the screen
- Components wrap up all of our styling and html for a single “widget”
- Components (except the app component) can be reused multiple times in an application
- Components can be nested inside other components
- The ‘App’ component is always the most parent component

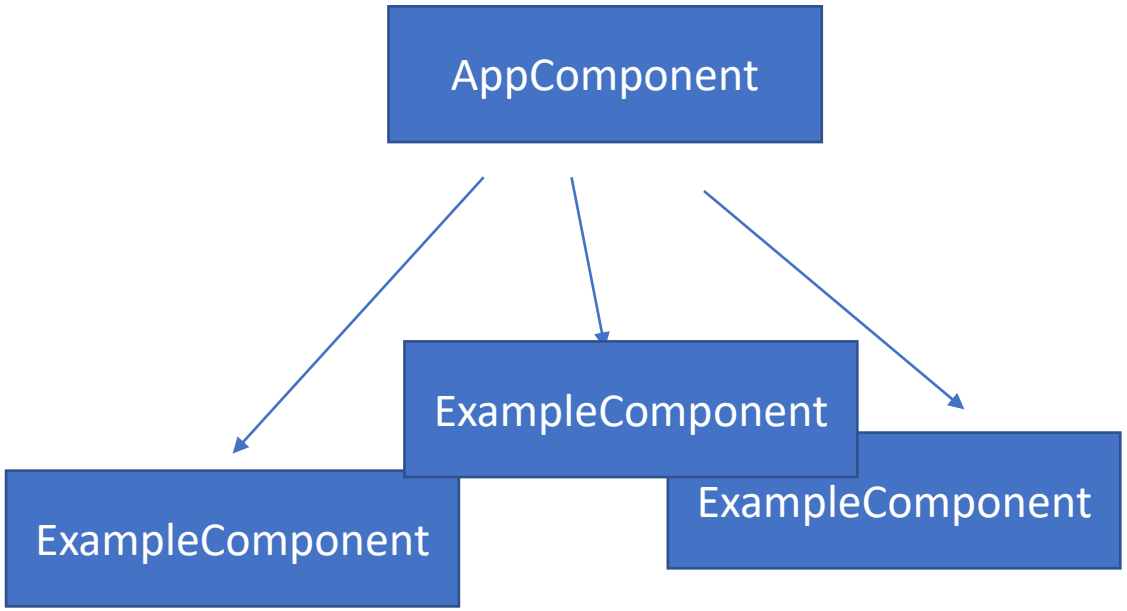
Component Files



- Each component has the following files:
  - <component-name>.component.html: contains the html for that particular component
  - <component-name>.component.css: contains the styling SPECIFICALLY for the html elements inside of the component template ONLY
  - <component-name>.component.ts: contains the TypeScript code linked to that component's behaviors and can be used to pass data between the template and actual programming logic
  - <component-name>.component.spec.ts: used for writing tests for our component

Component Hierarchy

- Whenever you build an Angular application, you should think about what components you want to create, and then create a diagram describing the “nested” structure
- App component is always at the top



Angular v. React

- Angular:
  - Considered a framework
  - Is more opinionated than React (meaning that Angular follows a certain structure of doing things much more strictly than React)
  - Created by Google
- React:
  - Considered a library (so just something you use with your plain JavaScript)
  - Is much less opinionated, more freedom to do different things you want to do
  - Create by Facebook