# Node js

* It is a free and open-source JavaScript interpreter/server environment (written in c++) that allows us to run JS outside of a web browser
* Platform independent
  + Meaning it can run on any OS
* Made from google chromium V8 engine

# NPM

* Node Package Manager
  + It is included with the download of Node.js
* Similar to Nuget Package Manager, it can manage our dependencies and versions and this is done through a **package.json**
  + **Package.json** – holds all the information of different packages and their versions that is needed for the angular application to run (you can check it out in your angular project and see just exactly what it stores)
* It will create a folder called **node\_modules** in our angular app
  + **Node\_modules** – important folder that actually stores the packages and their pre-made classes that we use in our angular app to do certain things
  + Ex:

RouterModule – lets us do routing in our angular app

HttpClientModule – lets us communicate with a rest api

ReactiveFormsModule – lets us create reactive forms in Angular

Etc.

## CLI

* Npm install -g typescript – install typescript globally for us
* Npm install -g @angular/cli – installs angular cli globally for us

# What is Typescript?

* It is an open-source language and **object-oriented** language developed by Microsoft
* Essentially, it made JavaScript into an actual object-oriented language
* It is superset of JavaScript
  + It has all the features of JavaScript but with more.
  + So, anything you find that is a JavaScript demo will most likely also work with your TypeScript file.
* It isn’t perfect and has problems with polymorphism (as shown in a demo)
* It is **strictly-typed**
  + Datatypes are enforced in typescript
  + Probably the most special thing with typescript

## DataTypes

* String
* Number
* Boolean
* Null
* Object
* Undefined
* Any – Default value to any variables that isn’t given a type and doesn’t hold any value
* Void – Used for functions
* Never – Used for exception handling

## CLI

* Tsc -v -will give you the current version of typescript
  + This might give you a problem if you didn’t set your execution policy to RemoteSigned
    - To set it up, open powershell as an administrator and run set-executionpolicy remotesigned and hit y for yes
* Tsc [filepath] – transpile the TS into JS
* Tsc [filepath] -w -will automatically transpile our TS into JS every time we save that file

# Angular

* It is an open-source framework TypeScript based front-end framework
* Angular first started as Angular JS (JavaScript based), but in Angular 2 they switched from JS to TS (**QC might ask this as “why did they change from JS to TS”**)
  + Angular JS did not support dynamic loading of the page or server-side programming language (Less speed)
  + Angular 2 better support for mobile devices
  + They changed architecture design **from MVC design to a Component and Directive design**
* It uses a component structure
  + A component is a single unit in Angular that displays some sort of visual to the user.
  + It is comprised of HTML, CSS, TS, and (optional) speck that represents a section/view of the application
  + Like an atom is the core component to make up something, a component is a core thing in Angular to make up a SPA or website

## Single Page Application (SPA)

* Like the name suggests, we only have a single page in our website
* The way it seems to “change a page” is that it will just replace the current component it is showing to a different component
* This capability allows us to navigate through our website without really refreshing the page at all and makes user experience a lot smoother

## Advantages

* It is more mobile friendly
  + Mostly because once it is loaded, the user can still navigate through the website without having to reload the page
  + Very useful for users with unstable internet connection (cellular network)
* Caching capabilities (it can store information in your system, so you don’t have to keep pulling it in)
  + It will synchronize your local data once connection is restored

## Disadvantages

* Doesn’t perform well with SEO (Search engine optimization)
  + Us human can perceive the webpage easily by the visual of the website, but a robot doesn’t see things visually, it can only see the files it retrieves and Angular uses JS to dynamically add and remove elements, so your html doesn’t really have information that displays the website unlike what you see.
  + You can test this by just looking at the page source of Angular doc website vs. the page source of the Wikipedia website.
* Initial load of the page might take a long time.
  + Since it needs to load all the components the app needs to display which can become a steep initial load at the beginning

## CLI

* Ng version – gives you the current version of your angular cli
* Ng new [projectName] – Creates a project for you
* Ng serve – Will run that angular application
* Ng generate component [Name] – Will create a component for you
* Ng generate service [Name] – Will create a service for you

# Angular Architecture

* A lot of information here So a couple tips:
  + Json files are usually used to configure something so Angular.json configures angular, package.json configures the packages
  + Anything in src folder is pretty good information to remember

## Node\_module

* Provides NPM packages for the entire workspace

## Package.json

* It tells us what our angular application needs from the node\_modules folder
* It also tells us the version of the package that we need

## Src Folder

* Contains the source files which gives information about the entire application logic, data, assets

## Assets

* Contains images, audio, video, or whatever external files you need to display your website

## Environment

* This folder contains information that your angular might need to reference on multiple times in your project

## Tsconfig.json

* It is the configuration to how you want your typescript to transpile or change certain properties of it.

## Main.ts

* The main entry point of the application like startup.cs in C#
* It specifies how to compile the application and specifies the main module.

## Index.html

* The main entry point of the view
* Adding any link tags here that references an external JS or CSS files will become available to all your components

## Angular.json

* Holds configuration information of how you want to run your angular app

## Package-lock.json

* Just locks the packages to a certain version and only use that version and no other version

# Decorators

* They are like annotations in C# in that they give extra information to whatever they are attached to.
* It is used to observation, modify, and replace definition

## Class decorator

* It is declared before a class declaration
* Ex: @NgModel, @Component, @Injectable, etc.

## Method decorators

* It is declared before a method declaration
* Ex: @HostListener (we never used this, so you don’t need to know what it does)

## Property decorators

* It is declared before a property declaration
* Ex: @Input, @Output (the two duos that is awesome to share information between a parent component and child/subcomponent)

# Directives

* **Don’t confuse with decorators**
* It allows us to manipulate the DOM
* It acts as a marker on the DOM element that tells Angular to change that DOM element either by appearance, behavior, and layout.
* Most directives will be indicated by “ng”
* You can also create your own directives in Angular
  + You must use the @Directive decorator

## Structural Directive

* Add or remove elements in a DOM
* Ex: \*ngIf, \*ngFor, and \*ngSwitch

## Attribute Directives

* It is used to change the look and behavior of the DOM
* Ex: ngClass or ngStyle

## Custom Directives

* You can make your own directives!

# Data Binding

* The process in where we share value within a component

## One-way data binding

* It is unidirectional

### Interpolation

* Allows us to bind data from the component to the view or the view to the component
* Syntax using “{{}}”

### Property Binding

* Syntax using “[]”
* It binds the attribute of an html element to variable in your TS

### Event Binding

* Syntax using “()”
* Binds the DOM events such as button click to a function in the component TS

## Two-way data binding

* Allows us to bind to data that will be shared to both the view and component bi-directional
* It essentially combines both property and event binding
* Syntax using “[()]”
* What it does is that it will listen for some event (Like what event binding does)and update the values simultaneously while that event is happening and you bind that value to an attribute of an html tag (like what property binding does)
* Very useful for child and parent components

# Pipes

* They provide a way to transform values before it is displayed
* If you have interpolation and want to change the look of the data it is showing, you use pipes!
* Syntax would be “{{targetDataYouWantToTransform | pipeType}}”
* Ex: someString | UpperCasePipe – will display “helloWorld” into “HELLOWORLD” in your view

# Services

* It is a class that is not dependent on any component
* They are used to share data/logic across components
* Services, if properly used, will be singletons
  + Meaning they will only have one instance always exist.
* It will injected into multiple components and those components can utilize the functions defined in the service class

## Angular Injector

* It implements dependency injection design pattern
* It just means you have to define in the constructor of the TS class what this class will need to function, and **Angular Injector** will handle giving that an object for you so you don’t have to do it.

## HttpClient

* It is the class that specializes in talking to different apis
* You need to import HttpClientModule in app.module.ts
* It returns an observable

# Routing

* A way for us to navigate through our website
* We define paths that when called upon will quickly swap the component that it is assigned to that path without refreshing the page
* It requires RouterModule in the app.module.ts file to work and <router-outlet> HTML tag

# Reactive Programming

* It is concerned with asynchronous programming that is pertaining to anything data/information being streamed/given to someone/something.
* Essentially, it is a library that has premade classes that are designed to deliver or retrieve data
* This is where observables come from particularly to **RxJs** library
  + So, if QC asks you what RxJs is, that’s a time to tell them about observables

## Observable

* It follows a publish and subscriber model
* It would continue to get data if they as subscribed to their endpoint
* It can have 0 or 1 or many activation
* It has an array that stores the data in the sequence they arrive in

### Publisher/Subscriber Design Pattern

* Describes the flow of messages between two entities
* A message is published by a **Publisher** to a **Channel**, any **Subscribers** monitoring that **Channel** will get notified and consume the data

A picture containing diagram

Description automatically generated

## Promises

* It can only have 0 or 1 activation so either it is successful or unsuccessful
* So unlike observables, it will not listen for more information and will just stop immediately after getting one response

# Angular Reactive Form

* All the form elements, user interactions, and validation are handled in the typescript file of the component.
* Easier to do more complex requirements of the forms
* You can make your own custom validation (but there are pre-made ones as well)

## Form-control validation states

* Form control has been visited
  + Meaning the user clicked on the input box
  + True – ng-touched
  + False – ng-untouched
* Form control’s value has been changed
  + Meaning the user typed some value in the input box
  + True – ng-dirty
  + False – ng-pristine
* Form control’s value is valid
  + Meaning what the user typed is not accepted
  + True – ng-invalid
  + False – ng-valid

# Basic understanding Testing in Angular

* We will just talk about the tools they use to do unit testing in Angular

## Jasmine

* It is a JavaScript unit testing framework that supports BDD
  + What is BDD?
  + Behavior Driven Development and it just means making test cases for user stories
  + Ex:

I want to create an account in this website => Create test case to check that making an account it possible => Create the function to do it

* + So, like TDD but more focus on laymen’s term kind of functionalities that you test for
* Tests will run in the browser and results are displayed in the browser

## Karma

* Manually running Jasmine tests by refreshing the browser or running it in different browsers each time you edit a unit test code gets extremely tiresome
* So, google developers created Karma to essentially automate that process
* It is a tool that will spawn browsers and run Jasmine tests all from the command line
* It can also display the results in the command line (but by default shows it in a browser)

## CLI

* Ng test