# Introduction to Node.js

* It is free and open-source JavaScript interpreter/server environment (written in c++) that allows us to run JS outside of a web browser.
* Platform independent
  + It can run on any OS
* Made from google chromium V8 engine with added event loops and low level I/O api.

## NPM

* It stands for Node Package Manager
  + It is included with the download of Node.js
* It consists of 3 components
  + Node.js, CLI, Remote Register
* Similar to Maven and similar to Nuget
* It will install our packages in a folder called node\_module our application

# CLI Command

* Node -v -Checks the version of Node
* Npm -v -checks the version of NPM
* Npm install -g typescript -will install typescript globally
* Npm install -g @angular/cli -will install angular CLI globally

# Introduction to TypeScript

* It is an open-source language developed by Microsoft
* It supports object-oriented programming language
  + Such as classes, interfaces, inheritance, etc.
  + Can’t do polymorphism that well
* It is a superset of JavaScript
* It is strictly typed
* Browser cannot interpret TypeScript

## CLI command

* Tsc -v -Checks the version of Typescript
* Tsc [Filepath] -Will transpile the typescript file into javascript file
* Tsc [Filepath] -w -Will transpile the typescript file into javascript file anytime there is a change in that typescript file.
* Tsc [Filepath] -w -t es[The year of the ECMA script] -Will transpile the typescript file into a newer version of javascript

## What else to research

* Interface
* Arrow Notation
* Modules – like packages in java and like namespace in C#

# Introduction to Angular

* It is an open-source framework TypeScript/JavaScript based front-end framework
* Angular first started as Angular JS (JavaScript based), but in Angular 2 they switched from JS to TS
  + Angular JS did not support dynamic loading of the page or server-side programming language
  + Angular 2 has better support for mobile devices
  + They changed architecture design from MVC design to a Component and Directive design
* Advantages
  + It was fast and responsive, it only updates the required components instead of the entire page.
  + Provides pleasant user experience on mobile devices, since it only changes the content and not the page
    - Especially when their internet speed isn’t reliable outside
  + Caching capabilities, since it retrieves everything at once and saves it to your device locally
* Disadvantages
  + Doesn’t perform well with SEO (Search Engine Optimization)
  + Initial loading takes awhile

## Webpack – bundling

* Loading multiple script tags into HTML pages is inefficient as it reduces the page speed since the browser must keep requesting each script file
* This can be solved by Bundling several files together into one file to be downloaded by the browser in one single request
* Webpack is a powerful static module bundled for JS application that packages all modules in our application and serves it
* Angular automatically does Webpack bunding in the background

## CLI Command

* Ng version – checks the angular version
* Ng new [project name] – creates the angular project

# Decorators

* Similar to Java notation, it adds special metadata to the class, function, parameters, (get/set), properties

## Class Decorator

* This is applied before the class.
* It is used to observe, modify, or replace class definition

## Method Decorators

* Applied before a method

## Property Decorators

* Applied before a property

## Accessor Decorators

* Applied before accessors

# Angular Directives

* It will allow 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

## Structural Directives

* Changes/Manipulates the structure of the DOM
* \*ngIf, \*ngFor, and \*ngSwitch

## Attribute Directives

* It is used to change the look and behavior of the DOM elements
* ngClass, ngStyle

# Data Binding

* The process in where we share/project values from the components TS file to the HTML file or vice versa
* There are different ways for data binding but we will focus on one-way or two way data binding

## One-way data binding

* Allows us to bind data from component to view or view to component
* It is only unidirectional

### Interpolation

* Syntax “{{}}”
* It will evaluate whatever is inside the expression in that way you can reference variables in the component in TS file or do some simple operations

### Property Binding

* Syntax “[]”
* It binds the attributes of the html element to the component’s state/variable

### Event Binding

* Syntax “()”
* Binds DOM events such as keystrokes, button clicks, mouse wheel to a function in the component

## Two-way Data Binding

* Allows us to bind data from both the component to the view and view to the component
* It is achieved by combining both property and event binding.
* Syntax “[()]”
* Angular uses ngModel directive
* Note: you must import FormsModule package in app.module.ts

# Component Lifecycle

* Each component is managed by the Angular framework
* Creates component, renders component, create/render any sub/children components
* Process changes to data bound properties
* Destroy the component

## Lifecycle hooks

1. Constructor is the first to get executed before any of the other life cycle hooks events
2. ngOnChanges() – called whenever one or more data bound properties change.
3. ngOnInit() – called once to initialize the component and set input properties.
4. ngDoCheck() – Called during all change-detection that Angular can’t detect on its own.
5. ngAfterContentInit() – Called once after Angular perform any content project into the component.
6. ngAfterContentChecked() – Called after each time Angular checks for content project into the component.
7. ngAfterViewInit() – Called once after Angular initialized the component’s view and child’s view.
8. ngAfterViewChecked() – called after each time Angular checks for the content projected into the component.
9. ngOnDestroy() – Called before Angular destroys the component or directive