**Angular**

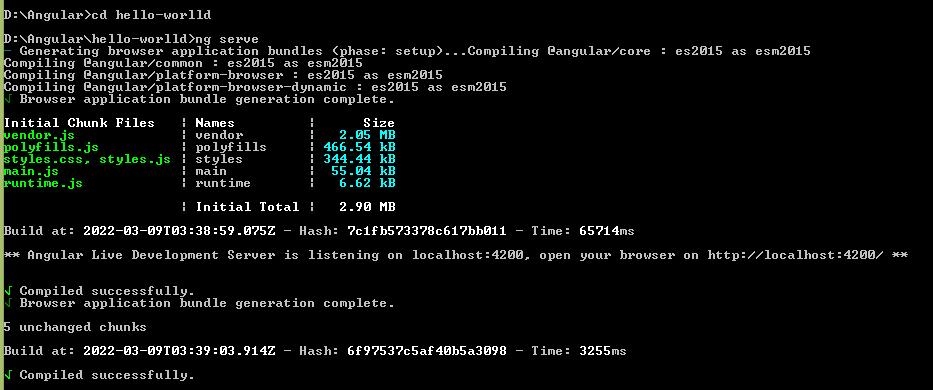
# Why Angular

* ***Frontend,*** (also called UI) is what we see on web page or website and validations to restrict or help for valid data. Built with HTML, CSS, java script (for validation and communicate with backend).
* ***Backend***, (what exactly business logic is and what it need to do) is what happened if a link is clicked or button is pressed or form is submitted, which is out of focus for website user. Several actions performed and respective intended operation completed like delete, create, update, fetch etc. Flow involves, backend validations, hitting data base (where all data resides).
* Any web application is combo of front end + backend. Many of existing old websites built using HTML + Java script or jQuery. If java script still existed what is need of building framework around it.
* Angular deals with front end, and just communicate with backend to pass request and get response from backend.
* Since websites are not stable; means, many functionalities or features added and removed as part of business requirement, maintenance is very difficult with plain vanilla java script.
* To make maintenance easy many java script design patterns (proven solution for particular kind of problem) used which makes life of developer’s worst ☺ and testing also tedious for large scale web applications with plain java script.
* So over past few years many java script frame works like Angular, React, Vue, expressjs etc. built and used to achieve above stated problems with large scale web applications.
* Framework gives advantage to application like structural, easy decoupling, and easy maintenance.

# Setting up development environment

* Mandatory is latest stable version of node JS(chose stable version not latest)
  + <https://nodejs.org/en/download/>
  + Why node required is, Node will make creating, running and deploying Angular apps super easy.
  + Once installed open command prompt and run
    - node --version
  + Node provides NPM (Node Package Manager) useful to install third party libraries.
    - Third party libraries like Angular, react etc.
  + We use Angular CLI (Command Line Interface) to create angular applications, and it reduce manual effort on creating so much of boiler plate code.
    - npm install -g @angular/cli
    - to make sure with installation ng --version

# Creating First Angular Project

* To create new project
  + Make sure the right folder selected to create code in CMD.
  + ng new hello-world
    - Takes more time to download required files. Around 15 to 20mins also based on Network speed ☹☹
  + So till now all skeleton code created and now need to add our own functionality on it.
  + We can start coding with note pad or word pad also.
  + But there exists more developer friendly code editors called IDEs.
  + One such IDE is Visual studio code.
    - <https://code.visualstudio.com/download>
  + Angular skeleton is always runnable without any issues. To run the application
    - ng serve
  + 
  + To see how web application looks like, go to browser type
    - localhost:4200/
    - If any change made to application in DEV those, the new changes will not reflect automatically and need to restart application every time.
    - Installing webpack will solve issue
      * **npm install -g @angular-devkit/build-web-pack**

# Project and folder structure

* Once project opened in Visual studio code, it organised in different folders
  + **e2e** End to End test folder
    - **but when I created application e2e folder not created ☹**
    - End to End testing is set of test cases which simulates actual user and test cases like
      * Opening code in browser
      * Clicking on a button
      * Fillout some form etc.
    - This folder contains 4 files
      * app.e2e-spec.ts
      * app.po.ts
      * tsconfig.e2e.json
  + **node\_modules**
    - node\_modules contains all the third party libraries on which the application depends on.
    - Remember, while deploying application node\_modules folder not going to be taken with deployment.
    - All the required dependencies and their versions bundled into package.json file and while installing application on server, the package.json file passed to some script and the script will read the file download required dependencies.
  + **Src**
    - Contains actual source code of angular application.
    - This folder contains other sub folers.
      * **App**
        + Contains 5 files

App.component.css

App.component.html

App.component.spec.ts

App.component.ts

App.module.ts

* + - * + UI of application is no single chunk, it is combination of different parts. Each logical part is called component. Also each functionality is also defined as components.
        + **Component.html** contains, logic that display data to user
        + **Component.ts** contains validation logic, backend connectivity logic etc.
        + **Component.spec.ts,** is for testing.
        + **Module.ts, will update ☺**
      * **Assets**
        + Assets needed for application like, image files, text files or icons will go into assets folder.
      * **Environments**
        + configuration settings for different environments

example, test environment will not use PROD DB server, not to be deployed to PROD application server, vice versa with PROD.

These setting s managed with configuration files

Environement.prod.ts 🡪 prod configurations go here

Environment.ts 🡪 DEV configurations go here

* + - * **Favicon.ico** 🡪 Icon to be displayed browser
      * **Index.html 🡪** contains angular application and displayed to browser
      * **Main.ts 🡪** starting point of application, likes boot strapping starting module of application.
      * **Polyfills.ts:**
        + Which contains some scripts that are required for running angular application
        + **Polyfills** fills gap between java script features required by angular application and the java script that supported by browser.
      * **Style.css** 🡪 styles required at application level.
      * **Test.ts 🡪** setting up test environement
      * **Angular.json 🡪** standard angular CLI configurations
      * **Editor.config 🡪**  for entire team, similar editor configurations to be use, those settings will be uploaded here.
      * **Gitignore🡪**
        + list of files to be ignored to push to git repository
      * **karma.conf.js 🡪**
        + configuration file for testing java script code
      * **package.json 🡪** determines list of files that application requires basically maintains dependencies list.
      * **Tsconfig.json 🡪**
        + contains settings for type script compiler.
        + Based on the settings typescript code compiled into java script code
      * **Tslint.json 🡪** static analysis tool for type script code. Checks readability, functionality and maintainability errors of typescript code based on this file.

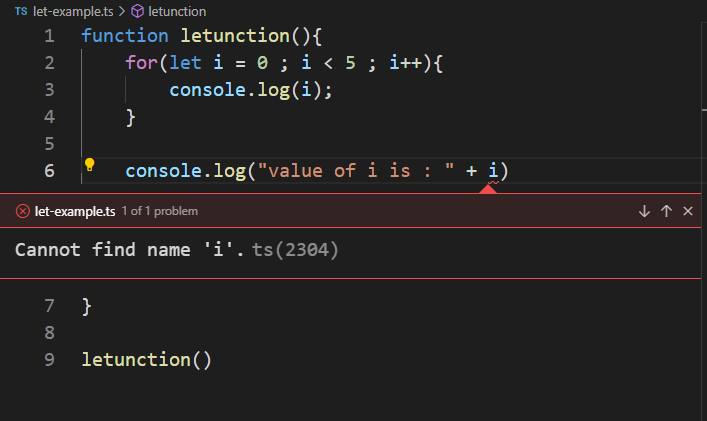
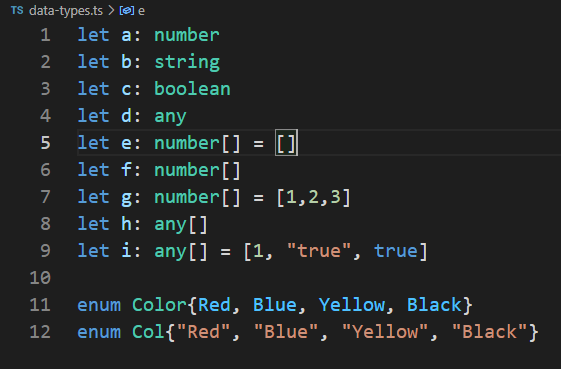
# Typescript fundamentals

* What is Typescript?
  + Superset of java script, which contains all JS features including additional features.
  + **Strong typing:**
    - Variable must tagged with data type.
    - This feature is optional in type script also but better to use
    - Object oriented features: Generics, access specifiers, constructors etc.
    - compile time errors
    - great tooling
  + Browsers can’t be able to understand type script so type script code need to transpiled (like compile) into java script, (it will happen automatically) while building application.
* **Installing type script**
  + npm install -g typescript
  + check version tsc --version
  + to compile
    - tsc file\_name.ts
  + new file will be generated as file\_name.js
    - function hello(name){

console.log(“Hello “ + name)

}

var some = hello(“Sunil”)

* to run generated JS file
  + node file\_name.js
* **var and let variables:**
  + the scope of ***var*** variable is scoped to nearest function
  + scope of ***let*** is limited to nearest block
  + 
  + here I is limited to for block and not accessible outside for block
* **data types in type script**
  + 
  + In enum, each value holds index value starting from 0, if any new value inserted between two values, index values get incremented
* **Type Assertions in type script**
  + Same as type casting, it will not change value of variable in memory; just tell compiler what type of data in variable.
  + 
* **Arrow functions:**
  + let something = (message) => console.log(message);
    - 