INTRODUCTION TO ANGULAR

## ANGULAR

Angular is a framework for building client applications in HTML,CSS and JavaScript/TypeScript.

Benefits of angular

* Gives our application a clean and loosely coupled structure.
* Includes a lot of reusable code
* Makes our application more testable.

## ARCHITECTURE

Data and Processing

User interface

BACK-END

FRONT- END

Data+APIs

## HTTP Services/APIs

End points that are accessible via the HTTP protocol.

## Node js

Run time environment for executing javaScript outside the browser.

Node – version gives the version of node js. Version 6.9 is the minimum version required for building angular applications.

Node Package Manager(npm) installs third party libraries.

Angular CLI is a cli tool to create new project or generate code or create deployable packages.

## SETTING UP EVELOPMENT ENVIRONMENT

npm install –g @angular/cli where g means global if we do not put g cli will be installed only in the current folder and it will not be accessible anywhere else.

Ng –version to know the version of cli.

It installs third party libraries and packages via npm.

To open project in visual studio

Shift+ctrl+p to open command palette 🡪 type code 🡪install code command in path”🡪.code in command prompt.

On giving ng serve , ng live development server is listening on localhost 4200, and angular cli also compiled our application.It generates bundles of javascripts and css files.

## STRUCTURE OF ANGULAR

* . e2e (end to end)-where we write end to end test. Automated tests that stimulate a real user. We can write code to launch our browser , navigate to home page of our application, fill a form, etc.
* Node\_modules –store all the third party libraries that our application may depend upon. When we run our application some of the libraries are put in a bundle and deployed in our application
* Src-source code of the application is present. Inside src we have
* app. Every app has atleast one module and a component.
* Assets- static assets for the application are stored, image file, text file, etc.
* Environments-configuration settings for different environments.

Environment.prod.ts-for production environment

Environment.ts for development environment

* Main.ts- typescript file is the sarting point of the application. we bootstrap the main module of the application i.e., the app module here. Angular loads this module and everything starts from there.
* Polyfills.ts fills the space between the JS feature needed by Angular and that supported by the browser.
* Style.css-to ad global style for application. each component has individual styles as well.
* Test.ts for setting up testing environment.
* Package.json-in dependencies we have libraries that our application is ependent on.devDependencies-dependencies needed to develop this application.(for developer machine)
* Tsconfig.json-settings for typescript and based on this typescript is converted to JS that browsers can understand.
* Tslint.json-settings fo tslint which is a static analysis code for TS.

### Webpack

Build automation tool. It get all our scripts and stylesheets combines them, put them in a bundle and minifies that bundle. This is for optimization.

Hot Module Replacement or Reloading(HMR)-Our webpage refresh automatically without refreshing on saving a source file.

Web pack injects scripts to index.html at runtime

# TYPESCRIPT

It is a superscript of javascript so any valid JS IS ALSO VALID ts CODE BUT ts but ts code has additional features that are not available in JS used by the browsers.

* Strong typing- We need not specify the variable type. However defining it helps at the time of debugging.
* OOP-
* Compile time errors
* Great tooling , intelligence available.

Transpile

JAVASCRIPT

TYPESCRIPT

When we give ng serve Angular cli calls the ts compiler to transpile all the ts codes under the hood.

#### Variable declaration in typescript

There are two ways to declare variables in typescript

1 var: in JS var is scope to the nearest function.

2 let is scope to the nearest block and not the function. This is much better.

#### Types in TS

1. Number
2. Boolean
3. String
4. Any
5. number[]
6. any[]

enum

#### Type assertion.

When we declare an instance variable without initializing its type is any . In that case intelligence wont be available. We can give assertions.

Message;

Let callmessage=(<string>message).function

Let callmessage=(message as string).function

#### Arrrow Function

JS FUNCTION: Let doLog=function(message){

Console.log(message)};

Using arrow function: let doLog=(message)=>{console.log(message)};

#### Interface

let drawpoint=(point{x:number,y:number}).This is called inline annotation .However this can be complex in the long run when a number of parameters are present. So we use interface. It increases reusability of the code. Interface are purely for declaration and cannot include any implementation. So this violates principle of cohesion .Hence we come to the concept of classes.

interface Point{

x:number;

y:number;

}

Interface follows Pascal naming convention.

## Class

Groups variables(Properties) and functions(methods) that are highly related.

class Point{

x:number;

fields

y:number;

drawPoint(){

/…

}

methods

drawDistance(another:Point){

/..

}

}

When dealing with object of a custom type we need to allocate memory to it.So we make use of new operator.

let point=new Point();

point.drawPoint();

### Constructors

In TS we do not have syntax for different types of constructor as in the case of java.So in order to accept a parametric constructor as non parametrized we can give it as follows

constructor(x?:number,y?:number){

this.x=x;

this.y=y;

}

### Access Modifiers

To control access to certain members of the class from the outside.

public

private

protected

In constructor we can prefix constructor with an access modifier. So TS compiler will generate these fields for us we don’t need to declare those variables. This reduces repetitive assignments.

constructor(private x?:number, privatey?:number){}

##### Property

It looks like a fields from the class but it is actually a method from inside example., getter and setter.

## Components

Building blocks of angular application. Component encapsulates the data, HTML markup and logic for view.

HTML TEMPLATE

LOGIC

DATA

1 Create component

@Component decorator converts a class into a component.

@Component(){}-decorator function takes in properties like selector, template, etc.

2 Register component to a module

@NgModule converts a plain TS file to a module from Angular’s point of view.In declarations we add all components that are part of the module.

3 call it in the app.component.html

ng g c component-name creates four file

.css for stylesheet

.html for template

.spec.ts for unit testing

.ts for class

## Module

A module is a container for group of related components. Organization of application into smaller functional areas. Whenever we have an import or export at the top of a file it is a module in TS’s point of view.

## Data Binding

Bind data to a view in the components.

## Services