**Command to install Angular Cli :** npm install -g @angular/cli@12.

**Command to uninstall Angular cli :** npm uninstall -g @angular/cli.

**Command to create new Angular project :** ng new ‘project name’.

**Angular project structure:-**

Node modules – this folder is that folder where all 3rd party library is stored where our project is dependent. This folder is for development purpose only not for production. When we are deploying our project we exclude this node module folder before deploy.

**SRC** – this folder contains all our source code.

Inside the source we have app folder.

**App** – when we are creating our project this app folder always got 2 things

1. App.componentt.ts //component
2. App.module.ts //module

Remember every anguler project must have one component and one module.

Whenever we want to to create a a new component or module or service we must create inside this app folder.

Assets – here we are storing static things like images, icons, text file etc.

**Environment :-** diff environment

Inside this file we have 2 enviroment by defult :-

1. Environment.prod.ts //for production enviroment
2. Environment.ts //for development enviroment

Inside this environment folder we can store environment configurations like database credencials or server ip addresses.

Index.html - this index.htm file is the main HTML file which gets rendered in the browser when Angular and runs if you notice here we don't have any references to any style sheet or JavaScript files this is because all these dependencies will be injected to this file during the build process by angular cli

**Main.ts** – this main.ts file is a type script file and this file is basically starting point of angular application. So in a lot of programming languages we have this concept of main method which is the starting point of a program and we have the same concept in angular as well so whenever this angular application will run the first file which will be executed is this main.ts file because this file is the entry point of any angular application in this file all we are doing is we are putting this main module to this project and here the main module is this app module so we are bootstrapping this app module to our project so when we will run this angular application angular will load this module and everything else will start from there

Style.css – here we are adding our global css for our application.

**editorconfig file** - so in this file we can specify a set of rules which every developer needs to follow while holding the application

**gitignore file** - in this file you can specify the files and folders which you want to exclude from your GIT repository

**angular.json** - this is one of the most important files and it contains all the configuration of our angular project so it contains configuration like what is the project name what is the root folder name what is the source folder name and other configurations

**package.json** - this is one important file this is a standard configuration files which every node project has in this file you can see we have some configurations like name of the project version of the project and other configurations then you will also see we have this dependencies and dependencies inside this dependencies you will see a list of dependencies on which our project is dependent

What is component?

a component is simply a typescript class decorated with @component decorator and it contains methods and properties which we can use in the view template of that component.

main component in any angular application is the app component and the app component is also called as the root component and you can say that it holds the entire application in the route component can have several nested components

@Component({

  selector: 'app-root',

  standalone: true,

  imports: [RouterOutlet],

  templateUrl: './app.component.html',

  styleUrl: './app.component.css'

})

export class AppComponent { //for this component the seclector is ‘app-root’

  title = 'first';

}

On the above code we have a AppComponent class , on that class we are adding @Component. @ is a decorator.

**Decprator (@)** - a decorator is metadata to the class making it an angular component.

Data binding : it allow us to to communicate between a component class and its corresponding view template.

One way data binding :

1. component -> view template through the string interpolation : {{data}} or property binding : [property] = data.
2. View template -> component class through Event binding : (data) = “expression”

Two way data binding is the combination of property binding and event binding.

**String Interpolation**

 slogan: string = "Hi I am Sayantan"

  remove(){

    return “this is sayantan”

  }

How to use this :

{{slogan}}

{{remove()}}

**Property binding :-**

By the property binding we are only binding properties like src , hidden etc

Src=”/gg/fr.jpg”

How to use this :-

<img [src]=”Src”>

**Event Binding :-**

export class NavComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

  slogan: string = "Hi I am Sayantan"

  scarch:string=""

  hello:boolean=false

  scarchvalue(eventdata:Event){

    this.scarch=(<HTMLInputElement>eventdata.target).value;

  }

  remove(){

    this.hello=true

  }

}

<p>nav works!</p>

<h1>{{slogan}}</h1>

Input : <input type="text"(input)="scarchvalue($event)">

<h1 [hidden]="hello">hi : {{scarch}}</h1>

<div>

    <button (click)="remove()">Remove</button>

</div>

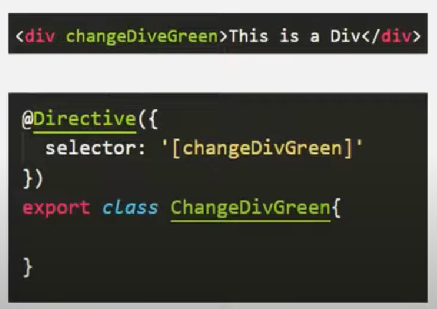
**Two way data binding:-**

We can achive this by property binding+event binding with the help of ngModel.

Directives :-

Directives are simply an instruction to the dom.

By default components are directives with the view template.

There is also another type of directive without view template.

Example:-

<div changeDiveGreen>This is green</div> //it will change color green for this div.

Note :- mostly we use directive as a HTML attribute.

now just like a component a directive is also a typescript class decorated with a directive decorator that means when we want to create a custom directive

remember that angular also provides us with some built-in directives like ngif ngfor ngstyle and the class etc

a directive can be classified into two types structural directive and attribute directive.

structural directive manipulates the top by adding or removing web page elements on the other hand the attribute directive is used like an attribute on an existing web page element and changes the look and behaviour of that particular HTML element does not add or remove elements from the web page it simply changes the look of the behaviour of the HTML element on which we are using that attribute

ngFor Directive :-

The ngFor directive is used to repeat a portion of HTML template once per each item from an itterable list.

The ngFor is a structural directive. That means, ngFor manipulates the DOM by adding or removing elements from the DOM.

ngFor directive Code:-

nav.component.ts

import { Component, OnInit } from '@angular/core';

@Component({

  selector: 'app-nav',

  templateUrl: './nav.component.html',

  styleUrls: ['./nav.component.css']

})

export class NavComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

  slogan=[

    {name:"Sayan",age:67},

    {name:"Megha",age:45}

  ]

}

Nav.component.html

<div \*ngFor="let item of slogan; let i =index">

    <div>

        <div>

            <p>Name is {{item.name}}</p>

        </div>

        <div>

            <p>AGE IS : {{item.age}}</p>

        </div>

    </div>

</div>

Science ngFor add HTML document to the webpage that’s why it is manipulating the dom. That’s why ngFor is a structural directives

ngStyle: it is the attribute directive

example :

<div \*ngFor="let item of slogan">

    <div>

        <div [ngStyle]="{color:item.name === 'Megha'?'Green':'red'}">

            <p >Name is {{item.name}}</p>

        </div>

        <div>

            <p>AGE IS : {{item.age}}</p>

        </div>

    </div>

</div>

ngIf :- it is a structural directive

* The ngIf directive is used to add or remove element from a webpage based on a given condition.
* If the condition assigned to ngIf returns true, it will add the element on which it is used to the webpage. Otherwise, if the condition returns false, it will remove that element from the webpage.

ngClass :-

* The ngClass is an attribute directive.
* The ngClass directive is used to add a CSS class dynamically to a webpage element.

Perent and child component :-

* We can pass data from component class to view template and vice versa using property binding, string interpolation and event binding.
* We can also pass data from parent component to child component and vice versa. We use @Input & @Output decorator for that.

We use @Input decorator to pass data from parent component class to child component class.

Remember to pass data form parent to child:-

1. Import input in chield ts file.