**Angular 18**

Go to console enter the command :-

**npm –v** (to check the version of node js )

To install the Angular CLI, open a terminal window and run the following command:

**npm install -g @angular/cli@17 ~ for version 17**

**npm install -g @angular/cli ~ for latest 18 or19**

next enter **ng v** = To get the Angular CLI version and other info.

* Open the folder where you have to create new project >
* Open command prompt on that folder and enter command
* Command for Creating new project :- **ng new < project-name >** > Press Enter key
* Select the CSS Framework
* Enter - No for server side rendering for new.

**ng serve –o =** for open project on browser.

**Angular Components :-**

* To create a new component – **ng generate component** **<component\_name >**
* Short command - **ng g c <name>**

Component – **Selector** = selector is is unique name for components. Whenever we have to use components then we can use with html tag of selector.

* Command for Run the project - **ng serve**
* Stop the compilation **– CTRL + C**
* We can use coponents in app.components.html by using component selector (in the form of Html tags**)**
* **First impost the component in app.component.html in imports**
* **To install Bootstrap – npn i bootstrap**
* Next add reference of link in **styles** of **angular.json –**
* **./node\_module/bootstrap/dist/css/bootstrap.min.css**

**Routing (navigating) in Angular :- ( in app.routes.ts )**

Export const routes : Routes = [

{

Path : “user-page”,

Components: **UserComponent** (component-name)

},

{

Path : admin,

Components : AdminComponents

}

]

**It will automatically imports the urls at top**

* **To enable routing in app.component.html we have to add <router-outlet> </router-outlet>**
* For navigation link in alternating of a href we have to use **routeLink=”/user-page” (component name)**
* **Using Constructor : (using ts file )**
* **Constructor(private router : Router){**
* **}**

**Navigate(){**

**This.router.navigateByUrl(“<component:selector>”}**

**}**

* **Via routerLink (via html) :- [routerLink ] =”[/component]”**

**Data Binding :-**

1. **One Way Data Binding :-**

**I. Via Inderpolation {{data}} =**

* **Interpolation** refers to **embedding expressions into marked up text**. By default, interpolation uses the **double curly braces** **{{ and }}** as delimiters.
* first create a variable add value that is to bind.
* And then where you have to bind data just add {{ data }} value will be bind.
* Ex- in app.ts = in class
* name ; string = “Jaydip”
* <h2> Hello {{Name }}</h2>
* That will show output Hello Jaydip

1. **Property Binding –**

* we can bind data in any html tag property.
* Ex - <p [class] = “clname”> This is Property Binding </p>
* We have to bind using [] square brackets.

1. **Event binding :-**

* In event binding first we have to crate an function.
* Attach to the html element like button , p , span, heading.
* Ex – greet (){

Alert(“ Hello, Welcome to my website “);

} // app.components.ts

<Button class=”btn btn-danger” (click) = “greet()”>Show message<button> //in app.components.html

1. **Two Way Binding :-**

* We can bind the data using [(ngModel)]

1. **Using Signal = signal() =**

**Directives :-**

1. **Structural Directive :-**
2. **\*ngIf** – this directive is used to insert or remove the element.

* We can remove or add html element on button click , dropdownselect or checkbox select.
* Jyala hide karayach aahe tyala \*ngIf vapara ani jyacha event var hide karayacha aahe ( button,checkbox or dropdown ) tyala [(ngModel )] vapara.
* To use the \*ngIf and \*ngFor we have to import the commonModule in component.ts file.

1. **\*ngFor –** It only needs an Array for itreration.

* **Syntax :**- <li \*ngFor=”let city of cityArray”>**{{city}}</**li>
* First create array which we have to itrerate.
* Then add the syntax to the element that which we use for itreration.
* Ex – we can use in <li> , cards <option > for multiple option in dropdown list.
* We can use \*ngIf inside \*ngFor for comparison.

1. **Attribute Directive :-**
2. **[ngClass] :**
3. **[ngStyle] :**

* **Control flow Statement :-**

1. **@if :-**

@if (istrue == true){

<div class=”bg-primary”>

Hello Jaydip </div>

@if(istrue){

<div class=”bg-primary”>

Hello Jaydip </div>

}

@else{

<div class=”bg-warning”>

Hello Jaydip </div>

}

1. **@for :-**
2. **@switch :-**

Daynumber :string = “”;

< input type= “text” [(ngModel)]=”daynumber”;

@switch(daynumber){

@case(‘1’){

<span> Monday </span>

}

@case(‘2’){

<span>Tuesday </span>

}

@case(‘3’){

<span> Wednesday </span>

}

@default{

<span> No day Selected </span>

}

}

* **Pipe in Angular :- (we need to import this every pipes)**

Angular provides built-in pipes for typical data transformations.

Datepipe, uppercasepipe, lowercasepipe, jsonpipe, etc.

Using pipe – (currdate | date : ‘dd-mm-yy’)

**Template Form Validation :**

**#hash-property =** it is a variable created in html. We can store the [(ngModel)] value.

* We can use control statement @if , @else.
* Ex - @if (variablename.errors ?. [‘required’]){
* This is required field.
* }

**Reactive form validation :-**

Reactive forms are defined in the component class using **FormGroup,** **FormControl,** and **FormArray**. Validation is applied directly in the component class using Angular’s **Validators** class.

Ex - myForm = new FormGroup({

name: new FormControl('', [Validators.required, Validators.minLength(3)]), }); }

**Get API using HttpClient :-**

* First create a **Service of HttpClient (**is .ts file**)**
* **Ex** - Constructor(private http :HttpClient){}

We can create object of httpClient using inject

* **Ex –** http = inject(HttpClient);

**Api Call :-**

# use subscribe for getting and storing data in variable.

Syntax : -

MethodName(){

this.http.get(“< - - API Url - - > ”).subscribe((res:any)=>{

})

**Services in Angular : -**

**services** are classes that are used to provide specific functionality that can be shared across different components of an application.

* Ex – Used for API calling.

**Reusable Components ;-**

**reusable components** are components that can be used in multiple parts of your application without needing to rewrite the same code.

**Input and Output Properties**: Components should expose @Input and @Output properties for dynamic data binding and communication with parent or child components.

* @Input: For receiving data from a parent component.
* @Output: For emitting events or notifying the parent component of changes.
* First we have to create separate folder for reusable components.
* Then create component in html