**Angular**

**Creating Project**

1.npm init @angular myApp

2.After creating a project check in vscode editor.

3.Can you run the project by using the command npm start.

4.for stop your project by using a ctrl + c (press y).to stop the project.

5.After running the project npm start.

**Installing Angular CLI:**

1.npm install -g @angular/cli

2.After run the command by using an ng

3.set the ng values **set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy. RemoteSinged.**

4.By using ng -- help give the details of the ng command process.

**Using Ng Command:**

1.After install the ng cli then by using an ng serve to run the project.

**Project Structure:**

1.src is the main page in the

index.html file as an important file.to run angular.this index.html has base html.

Style.css to handle the all style page.

2.main.ts

This will run your application run typescript.

3.Favicon.ico

Logo of image.

4.app

Main files has app.component.ts.this one of main components

Each components has main html.css properties

app.routes.ts has an url declared and rotes.url all files.

**Create a Hello word:**

@**Component**({

selector: 'app-root',

imports: [**RouterOutlet**],

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

template:"<h1>Hello world</h1>",

styleUrl: './app.component.css'

})

export class **AppComponent** {

title = 'myApp';

}

By using String interpolation example :Hello world.

@**Component**({

selector: 'app-root',

imports: [**RouterOutlet**],

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

template:"<h1>{{title }}</h1>",

styleUrl: './app.component.css'

})

export class **AppComponent** {

title = 'Hello world !';

}

**Fixing Missing app.modules.ts file.**

This app.modules now show an current version of angular project.

Now,Create standaloneComponents.

1.Where you stand alone an app.components.ts when used as standalone values.no need to create an individual module.

2.it will import all related components.

3.App not using any standalone as this and creating an app will cause a performance issue and file.

4.If can create non standaloneComponents.

By using an command as : **ng new myApp --no-standalone**

If using a standalone project by using an **ng new myApp --no-standalone** this command. Orelse an an normal command **ng new myApp.**

**Data Binding.**

Data binding in Angular is a mechanism that allows communication between the component and the view (HTML). It helps synchronize data between the TypeScript logic and the UI without manually updating the DOM.

Type of data Binding

1. one way Binding.

2.Two way Binding.

One way Binding:

**One-way binding in Angular** is a data binding technique where data flows in only one direction—either from the component to the view or from the view to the component.

**OneWay Data Binding Example:**

By using an oneway data binding an only communication data into view page into component

Example

@**Component**({

selector: 'app-root',

imports: [**RouterOutlet**],

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

styleUrl: './app.component.css'

})

export class **AppComponent** {

title = 'myApp';

}

In an view template url value has an './app.component.html',

<h1>{{title}}</h1>

It will show a chrome page.

Error

<h1>{{title= “Sometext”}}</h1>

Parser Error: Bindings cannot contain assignments at column 8 in [{{title= "Sometext"}}]

Note: In an oneway binding if an html file is not reflected in the component.only show any data in view page.

**Two Way Data Binding**

Two-way data binding in Angular allows synchronization of data between the component and the view. When the user updates the value in the UI, it automatically updates the component, and vice versa.

By using an Ngmodel value.

**Two Way Data Binding Example:**

@**Component**({

selector: 'app-root',

standalone: true,

imports: [**RouterOutlet**,**FormsModule**],

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

styleUrl: './app.component.css'

})

export class **AppComponent** {

fullName = "Testing Input ";

}

In an view template url value has an './app.component.html',

<h1>{{fullName}}</h1>

<input *type*="text" *[(ngModel)]* = "fullName"> //here,*ngModel for using an databinding.*

It will show a chrome page.

**String InterPolation**

**String Interpolation in Angular** is a data binding technique used to display data from the component in the view (HTML). It is done using **double curly braces {{ }}**.

If property value show in html value as been into show and values is known an string Interpolation.it allow an functions also

By using an Syntax : {{}} an show an value in the view page. {{expression}}

**String InterPolation Example:**

@**Component**({

selector: 'app-root',

standalone: true,

imports: [**RouterOutlet**,**FormsModule**],

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

styleUrl: './app.component.css'

})

export class **AppComponent** {

title = "Hello world"

**getName**(){

return "Angular Developer";

}

}

In an view template url value has an './app.component.html',

<h1>{{title}}</h1>

<h2>{{getName()}}</h2>

It will show a chrome page.

**Property Binding**

Property Binding is a data binding technique in Angular where data flows from the component to the view (HTML element property) using square brackets [].

Check DOM properties. It will show an all html properties

Example: <h1 [innerText] = “tittle”></h1>

|

This has properties

<button [disabled] = “isDisabled”>Click me! </button>.

|

This has properties

**Property Binding Example:**

@**Component**({

selector: 'app-root',

standalone: true,

imports: [**RouterOutlet**,**FormsModule**],

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

styleUrl: './app.component.css'

})

export class **AppComponent** {

title = "Hello world"

}

In a view template url value has an './app.component.html',

<h1 *[innerText]* = "title"></h1>

Event Binding

**Event Binding** is a data binding technique that allows communication from the **view (HTML)** to the **component (TypeScript)**. It is used to handle user interactions like clicks, keypresses, or input changes.

example : Component onsave()

<button (click) = “onsave()” > Save</button>

**Event Binding Example:**

@**Component**({

selector: 'app-root',

standalone: true,

imports: [**RouterOutlet**,**FormsModule**],

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

styleUrl: './app.component.css'

})

export class **AppComponent** {

clickcount = 0;

**clickme**(){

this.clickcount ++;

}

}

<h1{{clickcount}}> times</h1>

<button (click)= “**clickme()”> click me</button>**

Create a new project for an counter.

Now create an new component **ng generate component counter** by using this command

After then show an counter components in html value

* remove an app.component html.
* Add app.component.ts import the **counterComponent.**

here ,attached main value component of Angular file.

@**Component**({

selector: 'app-root',

imports: [**RouterOutlet**,**CounterComponent**],----> here import new component

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

styleUrl: './app.component.css'

})

export class **AppComponent** {

title = 'simplecount-app';

}

counter.component.ts:

@**Component**({

selector: 'app-counter',

imports: [],

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

styleUrl: './counter.component.css'

})

export class **CounterComponent** {

count = 0;

**increment**(){

this.count++;

}

**decrement**(){

if(this.count > 0){

this.count--;

}

}

}

Counter.component.html

<h1>Count : {{count}}</h1>

<button *(click)* = "increment()">+</button> &nbsp;

<button *(click)* = "decrement()">-</button>

**Directives**

A **directive** in Angular is a special instruction that **enhances HTML elements** by adding behavior, modifying structure, or styling elements dynamically.

### **Types of Directives:**

**Component Directives** – Custom elements with templates (@Component).

**Structural Directives** – Modify the DOM structure (\*ngIf, \*ngFor, \*ngSwitch).

**Attribute Directives** – Change element appearance/behavior ([ngClass], [ngStyle]).

**Structural Directives**

\*ngFor

@**Component**({

selector: 'app-root',

standalone: true,

imports: [**RouterOutlet**,**FormsModule**,**CommonModule**],

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

styleUrl: './app.component.css'

})

export class **AppComponent** {

movies = ['Zootopia','Batman Vs Superman','Harry Potter','X-men'];

heroes = ['Vijay','Ajith','Karthick',"Arunvijay"];

}

In a view template url value has an './app.component.html',

<h1>\*ngFor for example </h1>

<ul>

<li *\*ngFor* = "let movie of movies"> {{movie}}</li>

</ul>

<ul>

<li *\*ngFor* = "let hero of heroes">{{hero}}</li>

</ul>

\*ngIf

This a condition statement from based on working the ngif method.

@**Component**({

selector: 'app-root',

standalone: true,

imports: [**RouterOutlet**,**FormsModule**,**CommonModule**],

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

styleUrl: './app.component.css'

})

export class **AppComponent** {

showme = true;

}

In a view template url value has an './app.component.html',

<input *type*="checkbox" *[(ngModel)]*="showme">

<p *\*ngIf*="showme">can you see me </p>

**Attribute Directives**

[ngClass]

This a condition from based on working the ngClass method.values by using an ngClass.

@**Component**({

selector: 'app-root',

standalone: true,

imports: [**RouterOutlet**,**FormsModule**,**CommonModule**],

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

styleUrl: './app.component.css'

})

export class **AppComponent** {

applyred = false

}

In a view template url value has an './app.component.html',

Apply Red<input *type*="checkbox" *[(ngModel)]*="applyred">

<p *[ngClass]*="{'primary': applyred}"> Sampletext </p>

In a view template url value has an ‘./app.component.css’,

.primary{

color:red;

}

.big{

font-family: 'Courier New', Courier, monospace;

font-size: larger;

}

[ngStyle]

Ngstyle to apply a style in value by an object value.

@**Component**({

selector: 'app-root',

standalone: true,

imports: [**RouterOutlet**,**FormsModule**,**CommonModule**],

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

styleUrl: './app.component.css'

})

export class **AppComponent** {

}

In a view template url value has an './app.component.html',

<p *[ngStyle]*= "{'color':'red','font-size':'40px'}">Sample text</p>