Angular

🔥 ---------- Angular Project Structure ---------- 🔥

📂 my-angular-app/ → 🏠 Root folder
- 📦 node_modules/ → Dependencies installed via npm
- 📂 src/ → Source files
  - 🅰️ app/ → Main application folder
    - 🎨 components/ → UI components for building the interface
    - 🔌 services/ → Handles API calls & utility functions
    - 📜 models/ → Data models (interfaces/classes)
    - 🛡️ guards/ → Secure routes with authentication
    - 🔄 pipes/ → Custom data transformations
    - ✨ directives/ → Extend HTML with custom behaviors
    - 📌 pages/ → Feature-based page components
    - 🅰️ app.module.ts → Root Angular module
    - 🅰️ app.component.ts → Root component (entry point)
    - 🅰️ app-routing.module.ts → Defines application routes
  - 🎨 assets/ → Static files (images, styles, icons)
  - 🌍 environments/ → Environment-specific configurations
  - 🚀 main.ts → Application bootstrapping
  - 📜 index.html → Main HTML template
  - 🎨 styles.scss → Global styles for the app
- ⚙️ angular.json → Angular project settings
- 📦 package.json → Project dependencies & scripts
- 🎯 tsconfig.json → TypeScript configuration
- 📘 README.md → Project documentation & guidelines

display 1st change in Angular

---

step 1: go to app.component.ts

```
export class AppComponent {


  name1="Rahul";
  name2="Raj";

  x=10;
  y=20;
}
```

step 2: go to app.component.html

```
<h2>{{name1}}</h2>
<h2>{{name2}}</h2>
{{x+y}}
```

webpage: http://localhost:4200/

Rahul
Raj
30

NOTE : You cannot write variable directly inside class but you can write inside function like below

```
export class AppComponent {


  hello(){
var/let/const value=100;
}
}
```

Interpolation in Angular

- whatever we write in {{}} it is know as interpolation as I shown above
- you can display data from TS to HTML file
- execute JS code in HTML

in TS

```
export class AppComponent {

x=10;
y=20;

  name1="Rahul";
  name2="Raj";

}
```

in HTML

```
{{x==y}} //true
{{name1==name2}} //false
{{name1.toUpperCase()}}//RAHUL
{{"hello".toUpperCase()}}//HELLO
```

NOTE : but you cannot do other JS operations like increment/decrement initialization variable etc

Angular CLI (Command line interface)

which is help us todo work fast

- To use cli need to install ---> npm install -g @angular/cli
- like creating components ----> ng generate component login or ng g c login
- executing commands ----------->ng version , ng new my-app , ng help

Components in Angular

Components are nothing but it like small part/section of the project which you can use and re-use anywhere in the project

ex: car tyre is part of car

you can create **component** by using
> ng generate component login
or
> ng g c login

step 1: goto login.comonent.html

```
<p>login works!</p>
```

step 2: goto login.comonent.ts add LoginComponent in imports as shown below

```
@Component({
  selector: 'app-root',
  imports: [RouterOutlet,LoginComponent],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
```

step 3: go to app.component.html

note : app-login is present app.component.ts in sector properties i.e. selector: 'app-login' same name using in app.component.html

```
<app-login></app-login>
```

Custom Components in Angular

step 1 : create register.component.ts

```
import { Component } from "@angular/core";

@Component({
   selector:"app-register",
   template: `<h1>Hello, I'm Register</h1>`
})

export class RegisterComponent{

}
```

step 2: in app.component.ts add imports: [RegisterComponent]

step 3: in app.component.html add <app-register></app-register>

Data types in Angular

- name:string="Rahul"
- age:number=25
- dob:any="hi" or 25 or true

Event in Angular

- Click (click)="functionname()"
- Double Click (dblclick)="functionname()"
- Mouse Events (mouseover), (mouseout), (mousemove)
- Keyboard Events (keydown), (keyup), (keypress)
- Input Change (input)="functionname($event)"
- Focus & Blur (focus), (blur)
- Form Submit (ngSubmit)="functionname()"

get and set value using input field

step 1: app.component.html

```
<h1>Your name:{{display_Name}}</h1>
<input type="text" (input)="getValue($event)" value="{{display_Name}}"/>
<br>
<br>
<button (click)="displayName()">Get Name</button>
<button (click)="setName()">set Name</button>
```

step 2: app.component.ts

```
name="";
display_Name="";
email="";
getValue(value:Event){
this.name=(event?.target as HTMLInputElement).value
}
displayName(){
this.display_Name=this.name
}

setName(){
  this.display_Name="Rahul"
}
```

get value using template

step 1: app.component.html

```
<h1>Your email:{{email}}</h1>

<input type="text" value="{{email}}" placeholder="enter email id" #emailField/>
<1-- #emailField --it is template -->
<button (click)="getEmail(emailField.value)">Get email</button>
<button (click)="setEmail()">set email</button>
```

step 2: app.component.ts

```
getEmail(val:string){
  console.log(val);

this.email=val
}
setEmail(){
  this.email="Raj@gmail.com"
}
```

if-else in Angular

---

ex1:

step1: in .html

@if(display){

}

step2: .ts

display=true

ex2:

step1: .html

@if(x==20){

}

step2: .ts

x=20;

ex:3

step1: hide and show button in .hmtl

```
<button (click)="hide($event)">hide</button>
<button (click)="show($event)">show</button>


@if(display){
 <div style="background:red;width:200px;height:200px"></div>
}
```

step2: in .ts

```
 display=true;
 hide(event:Event){
 this.display=false

 }
 show(event:Event){
 this.display=true
 }
```

-------else---------

ex1:
step1: in .html

```
<button (click)="changeColor('red')">red</button>
<button (click)="changeColor('green')">green</button>
<button (click)="changeColor('yellow')">yellow</button>
<button (click)="changeColor('other')">other</button>


@if(colors=='red'){
   <div style="background-color: red;width: 200px;height: 200px;"></div>
}
@if(colors=='green'){
   <div style="background-color: green;width: 200px;height: 200px;"></div>

}
@if(colors=='yellow'){
   <div style="background-color:yellow;width: 200px;height: 200px;"></div>

}
@else if(colors=='other'){
   <div style="background-color: rgb(0, 0, 0);width: 200px;height: 200px;"></div>

}
```

step2: in .ts

```
colors=''
changeColor(s:string){
this.colors=s
}
```

switch in Angular

step1: in .html

```
<button (click)="changeColor('red')">red</button>
<button (click)="changeColor('green')">green</button>
<button (click)="changeColor('yellow')">yellow</button>
<button (click)="changeColor('other')">other</button>

<input type="text" (input)="changeColorByNumber($event)"placeholder="enter color name"/>

@switch(colors){
   @case('red')
   {
      <div style="background-color: red; width: 200px;height:200px;"></div>
   }
   @case('green'){
      <div style="background-color: green; width: 200px;height:200px;"></div>
   }
   @case('yellow'){
      <div style="background-color: yellow; width: 200px;height:200px;"></div>
   }
   @default{
      <div style="background-color: black; width: 200px;height:200px;"></div>
   }

}
```

step2: in .ts

```
 colors=''
 changeColor(s:string){
 this.colors=s
 }
 changeColorByNumber(event:Event){
  this.colors=(event.target as HTMLInputElement).value
```

for loop in Angular

step1: in .html

//note using track in for loop compulsory

```
@for(studens of studentsList;track studens){
<h1>{{studens.name}}{{studens.age}}{{studens.email}}</h1>

}
```

step2: in .ts

```
studentsList=
[
  { name:'Abhira',age:29,email:'Abhira@gmail.com'},
  { name:'Ruhi',age:26,email:'Ruhi@gmail.com'}
]
```