Angular

Angular Structure

display 1st change in Angular

Interpolation in Angular Angular CLI

(Command line interface)

Components in Angular Custom

Components in Angular Data types in

Angular Event in Angular get value

using template

if-else in Angular

switch in Angular

for loop in Angular

Signals

effect

@if condition

for loop contextual variable

Two way binding

Dynamic styling

Directives in 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){

<div style="background:red;width:200px;height:200px"></div>
}
```

step2: .ts

```
display=true
```

ex2:

step1: .html

```
@if(x==20){

 <div style="background:red;width:200px;height:200px"></div>
   }
```

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'}
]
```

Signals

without computed in .html

```
<h1>{{z()}}</h1>
<button (click)="updateValue()">update value</button>
```

step2: in .ts

```
  x=10
  y=20
  z=this.x + this.y

  updateValue(){
  console.log(this.z);------------->z=30
  this.x=50
  console.log(this.z);-------------->=30

  }
```

only you will get 30 output even tho your updating x=50

Signals

with computed in .html

```
<h1>{{z()}}</h1>
<button (click)="updateValue()">update value</button>
```

step2: in .ts

```
x=signal(10)
y=signal(20)
z=computed(()=> this.x()+this.y());
```

Signals

```
updateValue(){
console.log(this.z());          z=30
this.x.set(50)
console.log(this.z());          z=70

}
```

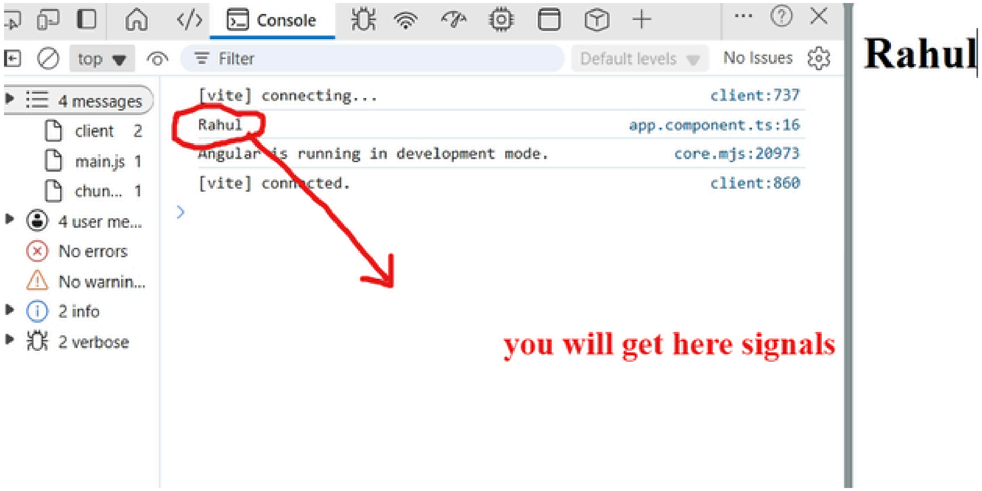
you will get 70 output because computed depencecies(binding x and y) so uou will get upadated i.e. x=50

effects

it is used when something is changed/updated you will get signal

step1: in .html

```
<h1>{{userName()}}</h1>
```



```
[vite] connecting...                                         client:737
Rahul                                              app.component.ts:16
Angular is running in development mode.             core.mjs:20973
[vite] connected.                                            client:860
```

Rahul

you will get here signals

step2: in .ts

```
userName=signal('Rahul')
constructor(){
console.log(this.userName());

}
```

or

```
userName=signal('Rahul')
constructor(){
effect(()=>{
console.log(this.userName());
})

}
```

with effect or without you will get signals because we are using constructor constructor will run on when application run you can set value here

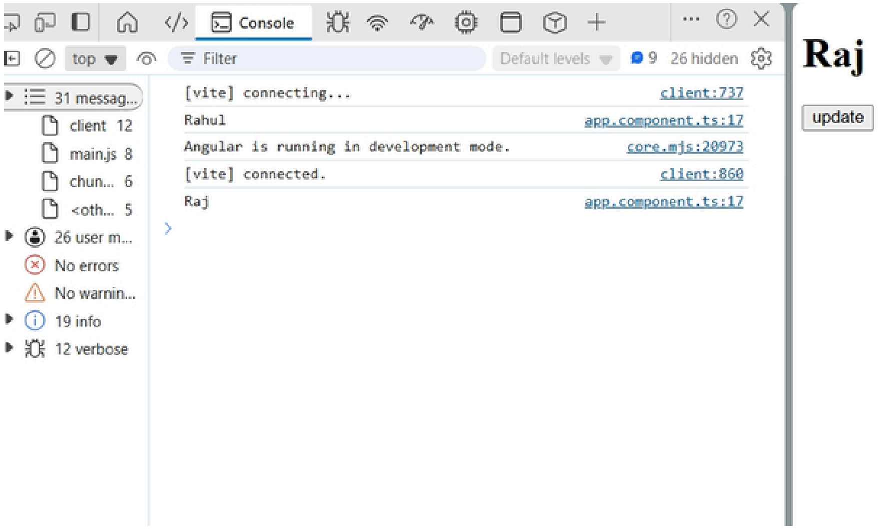
step1: in .html

```
<h1>{{userName()}}</h1>
<button (click)="userName.set('Raj')">update</button>
```

step1: in .ts

same .ts code as above used

```
 userName=signal('Rahul')
 constructor(){
effect(()=>{
 console.log(this.userName());
})


 }
```



```
[vite] connecting...                      client:737
Rahul                              app.component.ts:17
Angular is running in development mode.  core.mjs:20973
[vite] connected.                         client:860
Raj                                app.component.ts:17
```

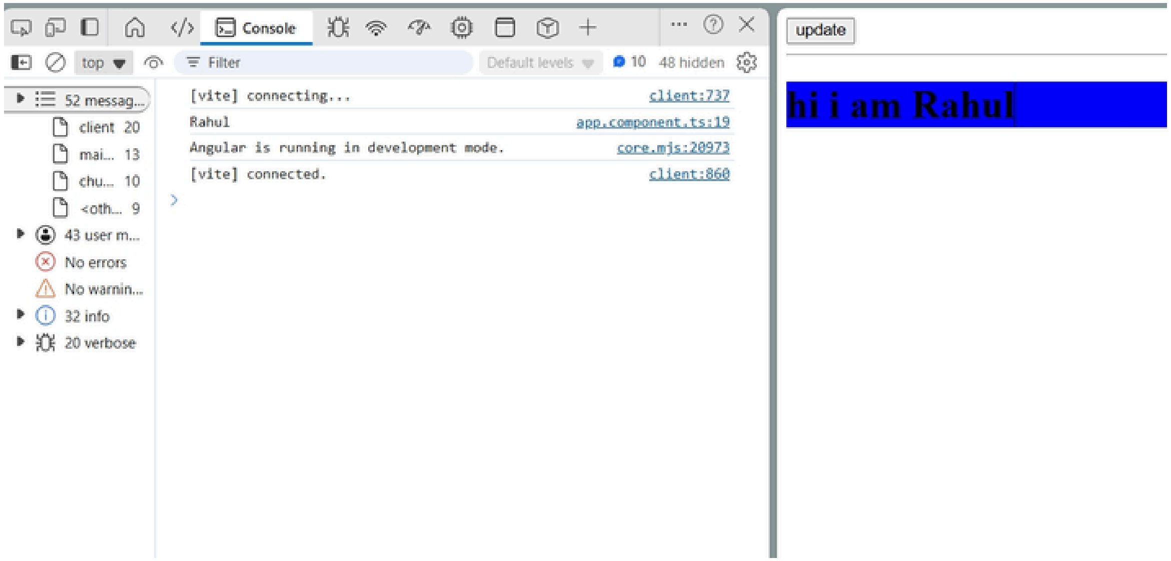
if display is true show text

step1: in .html

```
<button (click)="toggleValue()">update</button>
@if(display){
<h1 style="background-color: blue;">hi i am Rahul</h1>
}
```

step2: in .ts

```
display=false

 toggleValue(){
  this.display=true
 }
```



```
[vite] connecting...                                   client:737
Rahul                                         app.component.ts:19
Angular is running in development mode.          core.mjs:20973
[vite] connected.                                      client:860
```

hi i am Rahul

if display is true display text

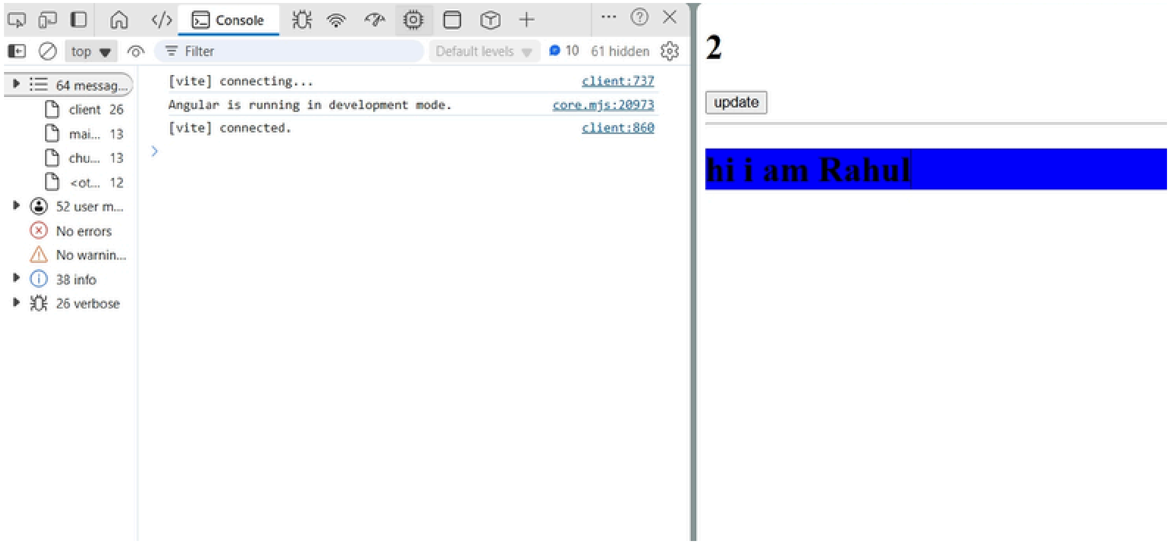
display text when count =2

step1: in .html

```
<h1>{{count()}}</h1>
<button (click)="toggleValue()">update</button>
@if(display){
<h1 style="background-color: blue;">hi i am Rahul</h1>
}
```

step2: in .ts

```
display=false
count=signal(0)

 userName=signal('Rahul')
 constructor(){
 effect(()=>{
 if(this.count()==2){
this.display=true
 }
 else{
  this.display=false
 }
 })

 }
 toggleValue(){
  this.count.set(this.count()+1)
 }
```



```
[vite] connecting...                              client:737
Angular is running in development mode.      core.mjs:20973
[vite] connected.                                 client:860
```

2

update

hi i am Rahul

remove text after 2 second just add below setTimeout code **in .ts**

```
display=false
count=signal(0)

 userName=signal('Rahul')
 constructor(){
 effect(()=>{
 if(this.count()==2){
this.display=true

setTimeout(()=>{
 this.display=false
},2000)
 }
 else{
  this.display=false
 }
 })

 }
 toggleValue(){
  this.count.set(this.count()+1)
 }
```

Data types in signals

step2: in .ts

```
y=signal<number|string>(20)
```

we can pass multiple data types in signals

```
 y=signal<number|string>(20)
```

we passing here number

```
  update(){
   this.y.set("Rahul")

  }
 }
```

we passing here string

step1: in .html

```
<h1>{{x}}</h1>

<button (click)="update()">update signal x</button>
```

Types of Signals

Writable signals

(WritableSignal<T>) are used for managing mutable state in Angular. They allow updates using .set(), .update(), and .mutate().

step1: in .html

```
<h1>{{count()}}</h1>
<button (click)="update()">update signal x</button>
```

step2: in .ts

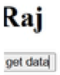
```
count: WritableSignal<number> = signal(0);
 update(){
  this.count.update(c=>c+1)
 }
```

computed signals

effect

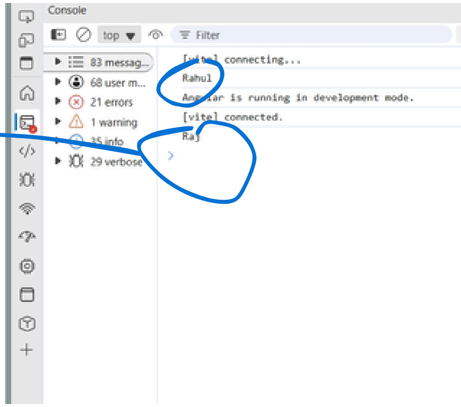
whatever you changeing in .ts it will give signals we use effect

```
<h1>{{username()}}</h1>
<button (click)="username.set('Raj')">get data</button>
```



Raj

```
username=signal("Rahul");

constructor(){
 effect(()=>{
  this.username()
  console.log(this.username());

 })
}
```



```
[vite] connecting...
Rahul
Angular is running in development mode.
[vite] connected.
Raj
```

getting signals when we use effect

@if()-condition

```
@if(displaying){
   <h1 style="background-color: red;">hello</h1>
}

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

```
displaying=false

show(Event:string){
this.displaying=true
}
hide(Event:string){
  this.displaying=false
  }
```

for loop contextual variable

in .html

ex1

```
@for(userdetails of users ; track userdetails){
<h1>{{$index+1}}{{userdetails}}</h1>
}
```

ex2

```
@for(userdetails of users ; track userdetails){
   <h1>{{$first}}</h1>
   <h1>{{$last}}</h1>
   <h1>{{$index+1}}{{userdetails}}</h1>
   }
```

ex3

```
@for(userdetails of users ;track userdetails){

    @if($even){
     <h1 style="background-color: aqua;">{{$index}}{{userdetails}}</h1>
    }
    @if($odd){
       <h1 style="background-color: rgb(242, 255, 0);">{{userdetails}}</h1>
    }
 }
```

in .ts

```
users=['Raj',"Rahul",'Ram','Abhira','Ruhi']
```

Two way binding

import FormsModule

```
<input [(ngModel)]="name" type="text" placeholder="enter name"/>
<h1>{{name}}</h1>
```

```
import { Component, effect, signal, WritableSignal } from
'@angular/core';
import { FormsModule } from '@angular/forms';
import { RouterOutlet } from '@angular/router';
@Component({
  selector: 'app-root',
  imports: [RouterOutlet,FormsModule],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {

name="Rahul"


}
```

import FormsModule to use two way binding

Dynamic style

in html

```
<h1 [style.backgroundColor]="bgColor"
[style.fontSize.px]="fontSizeSmall"
>hello i'm Rahul</h1>

<h1 [style.fontSize.px]="fontSizeBig">hello i'm Rahul</h1>
```

in .ts

```
bgColor="green"
fontSizeSmall="10"
fontSizeBig="90"
```

Directives in Angular

In Angular, directives are used to extend the functionality of HTML by attaching custom behaviors to elements in the DOM. Directives allow you to manipulate the DOM, apply conditional rendering, and reuse common functionality throughout your application.

Types of Directives

1️⃣ Structural Directives (Change Layout)

~~2️⃣~~ Attribute Directives (Change Look or Behavior)

~~3️⃣~~ Custom Directives (Your Own Magic!)

not required

1️⃣ Structural Directives (Change Layout)

*ngIf()

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

login=false

}
```

```
<h1 *ngIf="login">loged in</h1>
```

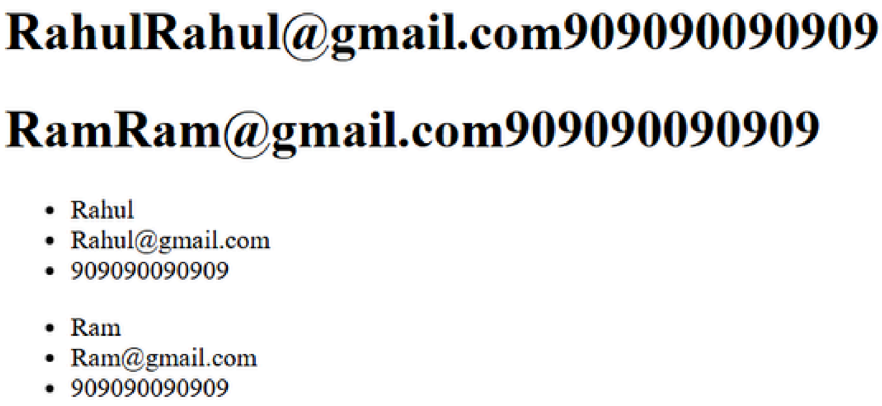
*ngFor()

```
@Component({
  selector: 'app-root',
  imports: [RouterOutlet,NgFor],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  cutomerList=[
    {
      name:"Rahul",
      email:"Rahul@gmail.com",
      mobile:909090090909
    },
    {
      name:"Ram",
      email:"Ram@gmail.com",
      mobile:909090090909
    }

  ]
}
```

```
<h1 *ngFor="let customers of cutomerList">
{{customers.name}}{{customers.email}}{{customers.mobile}}
</h1>
```

OR

```
<ul *ngFor="let customers of cutomerList">
<li>{{customers.name}}</li>
<li>{{customers.email}}</li>
<li>{{customers.mobile}}</li>

</ul>
```



```
RahulRahul@gmail.com909090090909

RamRam@gmail.com909090090909

  • Rahul
  • Rahul@gmail.com
  • 909090090909

  • Ram
  • Ram@gmail.com
  • 909090090909
```

ngSwitch

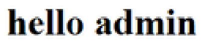
You need three parts:

1️⃣ ngSwitch – The main control (like a question).

2️⃣ *ngSwitchCase – Different options (like answers).

3️⃣ *ngSwitchDefault – A fallback (if none of the above match).

```
<div [ngSwitch]="userRole">
<h1 *ngSwitchCase="'admin'">hello admin</h1>
<h1 *ngSwitchCase="'user'"> hello user </h1>
<h1 *ngSwitchDefault>Invalid user</h1>
</div>
```



```
hello admin
```

```
@Component({
  selector: 'app-root',
  imports: [RouterOutlet,NgSwitch,NgSwitchCase,NgSwitchDefault],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
userRole="admin"

}
```

PART 1