Important topics: !

1.Modules

2.Components

3.Directives

4.Templates

5.Metadata

6.Pipes and filters

7.Routing

8.Services

9.Forms

10.Data Binding

11.Dependency Injection.

**video-5 Onwars**

in this class we discussed one new component selector how many ways to place in Root component,

that is reviewed in videoNumber-6

**videos-6 watching**

**Data Binding**

**Note:**

Data Binding allows us to communicate between component class and its corresponding view template.

2Types

1.One-way Data Binding.

1.String Interpolation (.ts 🡪 .html)

2.Property Binding (.ts 🡪 .html)

3.Class Binding (.ts 🡪 .html)

4.Style Binding (.ts 🡪 .html)

5.Event Binding (.html 🡪 .ts ) // this is reverse Binding.

2.Two-Way Data Binding:

ngModel (.html 🡨🡪 .ts)

Q) How many ways to place one new component in root component?

1. There are 3 way to call components

1.<app-sample></app-sample> //as a element

2.<div class=”app-sample”></div> //

3.<p app-sample></p> //

Every one using 1st one that is simple

**Interpolation** .ts => .html

**typescript file** we call **class file**

**html file** we call **view files**

Q) What is interpolation.?

a) Accessing class file property values to view files that is Interpolation using {{}}, And it will com view file to class file also.

Creating component

>ng g c components/simple --skipTests=true(presenter) (need to write version SpectFile Skiping command is changed)

**Sample.components.ts**

export class SampleComponnt implements OnInit{

public myName = “Smith”

}

**Sample.components.html**

<h2>This is Sample component </h2>

<h2> My Name is : {{myName}} </h2> //using interpolation accessing class file property values to view file

<h2>{{“My Name: ” +myName}} // in interpolation Strings appending

<h2>{{myName.length}} // we can find particular property length

<h2>{{myNam.toUpperCase()}} //we can use method in interpolation.-you will get all letters uppercase

<h2>{{myNam.toLowerCase()}} //you will get all property values will be lower case.

-----------------------------------------------------------------------------------------------------------------------------------------

**Sample.components.ts**

export class SampleComponnt implements OnInit{

public myName = “Smith”

showMyName(){

return “Hi Mr:”+this.myName; //because of local property write **this.**

}}

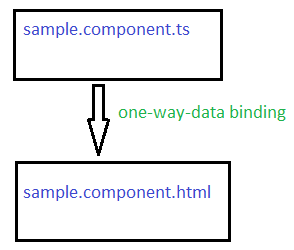
**Sample.components.html**

<h2>{{ showMyName()}}</h2> //calling method.

**Note:-**

1. Interpolation we can not add values.
2. In interpolation Global propertys directly we can not access. (Time:6:30)

By using interpolation we are creating **one way data binding**



**Property Binding:**

* To set, the property of an element in view to a property defined in the component’s class
* Property Vs Attribute:

1. Attribute and properties are not the same.
2. Attribute are defined by HTML.
3. Property are defined by DOM.
4. Attribute values can not be change once they are initialized.
5. Property values can change.

Attributs are different and Propertes are different property values in DOM, Attributs values in html, and html values can not changed but DOM values can able to change.So lets check below

For Property Bind we created one component

>ng g c components/p-Binding --skip-test=true (presenter)

Give property binding selector name to root component as a element **EX:** <app-p-binding></ app-p-binding>

**p-binding.component.html**

<h2>Property Binding</h2>

<input type=”text” value=”name”> // lets check values is attribute or property

Save and execute it.

In output goto input box right click select “inspect” and goto console on browser write command like

**Ex:**

**Checking as Attribute**

$0.getAttribut(‘value’) you will get output is 🡪name

**Checking as property value**

$0.value (presenter) you will get output is 🡪name

So it we check attribute getting same value and checking as property also getting same value .

So “**name”**  attribute come from **html** so it will **not change**

And output box I am changing value **name** to **age**

and check again attribute in browser console

EX: $.0getAttribete(‘value’) // 🡪 output is still **name**

So we did the changes on DOM (means browser) we will check dom values on browser console.

EX:$.0value (presenter) // 🡪output is **age**

**EX:-2**

**p-binding.component.ts //**our concepet is class property value need to be bind on view file.

export class PBindingComponnt {

public name= “smith”

public isDisabled : boolean = false

}

**p-binding.component.html**

**//** Binding with class property by using **[ ]**

**//**Ex:  **[value] =”name”**

<input type = “text” **[value] =”name”**> //

<input type=”text” value={{name}}>

**//by using interpolation also we did same thing so why property binding Why because in interpolation Boolean values are not work. That’s why using property binding.**

//lets check interpolation is work not in Boolean //by default Boolean value is true so I am adding one Boolean property on class file

<input type=”text” [value]=”name” disabled> // if you give disabled in prop

Lets check as interpolation on below

<input type="text" value="name" **id**="name"> //in this line why we use **id** check in internat.

<input type="text" value={{name}} id={{name}}>

<input type="text" [value]="name" [id]="name">

<hr>

<input type="text" [value]="name" [id]="name" disabled>

<input type="text" [value]="name" [id] = "name" disabled = {{isDisabled}}>

<input type="text" [value]="name" [id] = "name" [disabled] ="isDisabled">

// in view file witch class property wants to bind a

**ClassBinding**

Another way on oneway data binding is class binding.

Data flowing from **Class file** 🡪 to **view file**

For that creating component.

>ng g c components/classBinding --skipTests=true (presenter)

Give class-binding selector name to root component as a element **EX:** <app-class-binding></ app-class-binding>

**Class-binding.component.html**

<h2> Welcome to class Binding</h2>

<h2 class=”**textColor**”>Hello Angular</h2> //**1.this is Regular format**

**Class-binding.component.css**

**.textColor{ //2.this is Ragular format.**

Color:red;

}

**Class-Binding-flow**



**//**Property value assigning

**class binding with specific style**.

**Class-binding.component.html**

<h2 **[class.textColor]=”true”>**Welcome Angular</h2> //classbinding with specific style using textColor from css

<h2 **[class.textSize] = “false”**>Welcome Angular</h2> //you will get normal value why because of writed **false**.

**Note**:- directly we can not use **true** or **false,**  take property form class file

<h2 [class.textSize]= “**required”>**Welcome Angular</h2> // reaurired add from class file

**Class-binding.component.css**

**.textColor**{

Color: red; }

.**textSize**{

Font-size: 50px; }

**Class-binding.component.ts**

Export class classbinding implements OnInit{

Public **required**: boolean=ture //palace this **required** in view file as class binding.

}

**Multiple css class Binding**

**Q)Where you need to Bind in**

**a) class file section we will group.**

**Class-binding.component.ts**

Export class classbinding implements OnInit{

Public **required**: boolean=ture

Public **stryleGroup** ={ //in css multiple class inside of class we are defining as a group property //this file we are using in view file

“textColor”: this.required, //must be write your css class.

“textSize”: this.required,

“textStyle”: **!** this.required //for example we don’t want this style use this **!**  not symbal before this it will not work.

}

}

**Class-binding.component.html**

<h2 **[ngClass]**=” **stryleGroup**“>Welcome Angular</h2> //we are binding **Group of class** so Write [**ngClass**] take value form class file **stryleGroup**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**(Videos-7)**

**StyleBinding**

This is similar to class binding

This is also OneWayDatabinding data flowing from **class file** to 🡪 **view file**

For this style binding we are creating new component

>ng g c components/styleBinding --skipTests=true (presenter)

Give style-binding selector name to root component as a element **Ex:** <app-style-binding></app-style-binding>

**Ex:1**

**Style-binding.component.html**

<h2> Style Binding</h2>

<h2 **[style.color]=”red”**>Wecome Angular</h2> //individual different styles we can dind. // if you give **red** you will get error why because Angular treated as **red** is **property** it will check in class file, So if you want **property** become a **value** give single quotes along with double quotes **‘ ‘** EX: **“ ‘** red **‘ “**  then it will work.

<h2 **[style.color]=” ‘ red ‘ ”**>Welcome Angular</h2> // it will work.

**Note**: if you provide value write double quotes single quotes, **EX**:- **“ ‘ red ‘ “**

If you are provide value write double quotes it will work, **EX**:- **“ myColor ”**

**EX:2 we are taking class property (like variable)**

**Style-binding.component.ts**

Export class styleBinding {

Public **myColor** =”**blue**” //declaring property in class file and give property name in view file it will work,

Public **myStyle** =”italic”

}

**Style-binding.component.html**

<h2 **[style.color]=” myColor ”**>Welcome Angular</h2> // here only provided double quotes it is working, why because **myColor** is property in class file.

<h2 **[style.color]=”myStyle”**>Welcome </h2>

**Ex:3 Turnery Operators.(conditional base also use Style binding)**

**Style-binding.component.ts**

Export class styleBinding {

Public **required**: Boolean = true;

}

**Style-binding.component.html**

<h2 [style.color]=”**required** ? **‘green’** : **‘red’** “>Welcome to Angular</h2> //using turnery operator.

//our condition is **true** display **green** condition **false** display **red** color we are providing **value** so give **single quotes ‘ ‘**.

**EX:4 Multiple styles Binding**

**Style-binding.component.ts**

Export class styleBinding {

Public **myStyleGroup** = {

Color: ‘red’,

fontStyle: ‘italic’,

fontSize: ’50px’

}

}

**Style-binding.component.html**

<h2 **[ngStyle]= “myStyleGroup”** >Group of Style Binding</h2> //grop of styles binding so using **[ngStyle]**

**Time 13:00**

**EventBinding**

this is also OneWayDatabinding but this is Rivers flow, it will come from **view file 🡪** to **class file**

for this EventBinding Creating Component:

>ng g c components/eventBinding --skipTests=true (presenter)

Give event-binding selector name to root component as a element **EX:** <app-event-binding></app-event-binding>

**event-binding.component.html**

<button type=”button” **(click)** = **”onClick()”** >Click Me</button> //event raicing Means if I click the button control goes to the class file,

// **(click)** is a **event raicing** and **onClick()** is **event handler,** // this **onClick()** is define in class file there write sum console message.

<h2>{{myMsg}} </h2> //he is added interpolation but you need check without this line out put come browser not. And this is not there in Folw diagram. **Time: 20:00**

**event-binding.component.ts**

export class EventBindingComponent {

public **myMsg** =’ ‘ //taking empty space and using this property name in **onClick()** method

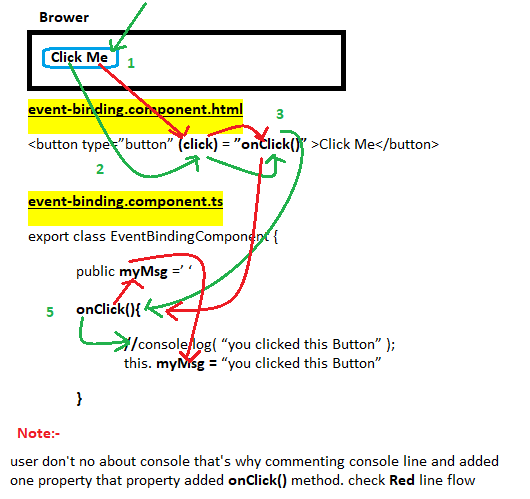
**onClick(){**

**//**console.log( “you clicked this Button” ); //user don’t no about console so commenting this line and displaying on browser writed below line.

this. **myMsg =** “you clicked this Button”

**}**

}//save it check out put on Browser Console. But user don’t no about console so we are commenting console line taking one property



**Brakates for usage**

**{} – Method Defination**

**{{}} – Interpolation**

**[] – property Binding**

**() – Event Binding**

**Note:- //what is property Binding? (answer check again)**

1. **Property binding noting but classbinding and even style Binding.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Two-Way-Data Binding**  **(Videos-8)**

**Installing bootstrap,**

**There are 2 ways to use Bootstrap**

1st option is Goto the browser there search boostrap5 you will find **CSS** link copy and goto the **indx.html** past it in **End** of the **Head section** //don’t use this link why because in future any updates will come then that time incompatibility issues will come that’s **why use second Option.**

2nd goto Terminal write command

>npm install bootstrap jquery --save (presenter)

Ofter installing Bootstrap it will not work give path of it,

Goto the Angular.json there give "Script[]" and "style[]" location

"styles": [

"node\_modules/bootstrap/dist/css/bootstrap.min.css"

],

“scripts”: [

“node\_modules/jquery/dist/jquery.min.js”,

node\_modules/bootstrap/dist/js/bootstrap.min.js"

]

First give jquery path and give bootstrap path remember it.

Save it, Stop the server (ctrl + c ) and start again (ng serve) it will work.

**event.binding.component.html:**

<button **class=”btn btn-success”**>Bootstrap Button</button> //this is normal button we are converting Bootstrap button so Appling **class=”btn btn-success”** you will get green color Bootstrap button

**Installing – AngularMeterial**

Open Angular material website link is **material.angular.io**  open it select **Material** you will find below link copy and past it in terminal and press entr.

>ng add @angular/material (presenter)

1. It will ask compatibility: (Press Yes)

2. It will ask colors: select what color you want by default it will show Indigo/Pink (press Yes) select it, if you want you can change later also

3. Angular Material typography (Press Yes)

4. Index.html also updated, So go to the index.html in body section please remove Class=”mat-typography” why because Angular Material typography Override to the Bootstrap so remove it.

**event.binding.component.html:**

<button>Material Button</button> //this is normal button this button we are converting to **Material Button** in below

For AngularMaterial purpose we are creating a Module

Time :13:00 getting sum confusion visit again **Two-Way-Data binding**

**(**

**This is line form project command under Modules folder AngularMaterial is created.**

>ng g m modules/angularMaterial (presenter)

**) // I implemented in below same.**

**Creating Module**

>ng g m module (pressEntr)

Open Module and Remove sum un wanted codes.

**Import {NgModule } from ‘@angular/core’;**

Import {comonModule} from ‘@angular/common’; Remove it

Import { MatButtonModul } from ‘@angular/matrial/button’; //material button added.

**Const materialComponents** =[ //material component purpose creating **Const[]** // go 1st Red color 2nd Yellow color 3rd Green color

MatButtonModul // this is button module

]

declaretions: [], Removie it

**imports:**[

ComonModules Remove it

**materialComponents**

],

**exports**:[ **materialComponents** ] Add exports

export class AngularMaterialModule { } // add this AngularMaterialModule class name in **app.module** (I think this is Root module) under imports section.

**Open app.module**

Imports :[

AngularMaterialModule// taken from angular-material.modules.ts class Go with Sagition it will automatically imported path.

**]**

We are using angular Material button so keep one tab on browser angular Material page.

Goto angular Material click the **component** and select **button** and copy **API** and past it we created new module check above **green** color imports

**event.binding.component.html:**

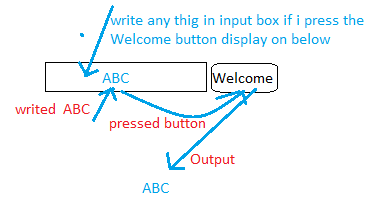
above we discussed we are converted normal button to material button we will see there.

<button>Material Button</button> //normal button

<button **mat-raised-button color=”warn”**>Material Button</button> //this is angularMaterial button why because we added this (fetchers or line) **mat-raised-button color=”warn”**

**Comment** above two buttons those are testing purpose.

Output came, but our requirement is write any thing if press the Welcome button that text need to display below.



<**input** type=”text” **#myMsg**> //view element capturing purpose use **Referenc\_Name** how to use Reference\_name by using **#myMsg**

<**button** mat-raised-button color=”primary” **(click)=”showMessage(myMsg)”** >Welcome</**button**> //normal button to material button converted. // click Event providing **Ex:- (click)=”showMessage()” //** input Reference\_Name providing to click event. That is **myMsg,** // and this **showMessage(myMsg)** define in EvntComponentBinding

**Export class EventComponentBinding{**

**showMessage(myMsg){**

**}**

}

**EX:2**

.html

<input type="text>

<button [(ngModel)] = "data" >Click me</button> //[(ngModel)] Two way data dinding purpose //data variabl from class file

{{data}} // this data from class file

.ts

public data = " "; // this date add in view section

// it will not work, why bacause they don't no what is ngModel so import FormsModule from app.module and declar import section.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**(Video-9)**

**Struchral Directives:**

**ngIf**

**ngSwitch //** ngSwitch and ngIf are conditional directives,

**ngFor //** list of item or loop purpose using.

**Creating component for NgIF**

>ng g c components/ngIf --skipTests=true (presenter)

Give ngIf selector name to root component as a element **Ex:** <app-ng-if> </app-ng-if>

**Note**:(Q) how to write ngIf (A)if we want write ngIf “**\***” is prefix and for **example** **\*ngIf =”true”** (write condition on true place)

**ng-if.component.html**

<h2>ngIf</h2>

<p **\*ngIf=”false”**>Lorem </p> // ngIf condition is false so this tag will not display.

<h2 **\*ngIf=”true”**>Lorem </h2> //this element is displayed why because condition is “**true**”

Save it check it, //and don’t use Boolean values directly take property name form class file.

<h2 \*ngIf=”**myStatus**”> Welcome to angular</h2>//take property name from class file.

**ng-if.component.ts**

**export class NgIfComponent {**

public **myStatus** = ture //give this property name to view file.

**}**

**Note:**- <ng-template> //ng template is special element this is else block rendring purpose this will use.

**EX: - 2**

**ng-if.component.html**

<h2>ngIf</h2>

<h2 \*ngIf=”myStatus; else **#block1**”>True Block</h2> //if you want particular element value use **Reference\_name** that is for EX: **#block1**

<ng-template **#block1**> // adding **Reference\_name** that is **#block1**

<h2>False Block 1</h2>

</ng-template>

<ng-template #**block2**>

<h2>False Block 2</h2>

</ng-template>

**ng-if.component.ts**

**export class NgIfComponent {**

public **myStatus** = false //give this property name to view file.

**}**

**EX: - 3**

**ng-if.component.html**

<h2>ngIf</h2>

<h2 \*ngIf=”myStatus; **then** **trueBlockelse1** **#block1**”></h2> //im not writing any thing h2 tag but I want group of statement should be run.// if the value is ture then displaying true block why because used **then trueBlockelse1**

<ng-template **#block1**> // adding **Reference\_name** that is **#block1** take any name your choice.

<h2>False Block 1</h2>

</ng-template>

<ng-template #**block2**>

<h2>False Block 2</h2>

</ng-template>

<ng-template #**trueBlock**1> // **Reference\_name** taking sum randam name

<h2>True Block 1</h2>

</ng-template>

<ng-template #**trueBlock**2>

<h2>True Block 2</h2>

</ng-template>

**Creating component for ngSwitch**

>ng g c components/ngSwitch --skipTests=true (presenter)

Give ngSwitch selector name to root component as a element **Ex:** <app-ng-switch> </app-ng-switch>

**Note**: Group of statement Execution purpose using **ngSwitch**

**ng-switch.component.html**

<h2>ngSwitch</h2>

<div **[ngSwitch]** =”**myChoice**”> // **[ngSwitch]** = binding with class property // **myChoice** Binding with class property

<div \*ngSwitchCase = “ ‘one’ ”> //in switchCase must be use “ \* ”

<h2>Case 1 </h2>

</div>

<div \*ngSwitchCase = “ ’two’ ”> //

<h2>Case 2 </h2>

</div>

<div \*ngSwitchDefault>

<h2>Wrong Choice</h2>

</div>

</div>

**ng-switch.component.ts**

export class NgSwatchComponent {

**myChoice** = “two” // depending on property value SwitchCase will print.

Check above line and comment it and use below line,

**myChoice** = “six” // this property value **SIX** is not there in view file then auto matically default file will b executed.

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**(video-10)**

**ngFor**

creating component for ngFor

>ng g c components/ngFor --skipTests=true (presenter)

Give ngFor selector name to root component as a element **Ex:** <app-ng-for > </app-ng-for>

**Note**: why we use ngFor to display list of items using **ngFor**

**ng-for.component.html**

<h1>ngFor</h1>

<div **\*ngFor = “let X** of **myName**; **index** as **i**“> //**X** is variable it is values bring from class property[], and you can take any name of variable we taken as **X** // **myName is from** class file property.

<h2>{{**i**}} - {{ **X** }}</h2> //**i**  is for index value it will look like 0123 like that.

</div>

Save it and execute.

Comment above div tag and execute below dive tag code.

**EX:2**

<div **\*ngFor = “let x** of **myName**; **first** as fir **”**> //capturing first element

<h2>{{ fir }} - {{x}} </h2>

</div>

Save it and execute.

Comment above div tag and execute below dive tag code.

**Ex:3**

<div **\*ngFor = “let x** of **myName**; **last** as la”> //capturing last element

<h2>{{ la }} - {{x}} </h2>

</div>

Save it and execute.

Comment above div tag and execute below dive tag code.

**Ex:4**

<div **\*ngFor = “let x** of **myName**; **even** as ev”> //even numbers only **true** this element

<h2>{{ ev }} - {{x}} </h2>

</div>

Save it and execute.

Comment above div tag and execute below dive tag code.

**Ex:5**

<div **\*ngFor = “let x** of **myName**; **odd** as od”> //odd numbers only **true** this element

<h2>{{ od }} - {{x}} </h2>

</div>

Save it and execute.

Comment above div tag and execute below dive tag code.

**Object also Displaying : \*ngFor**

**Ex:1**

**ng-for.component.html calss file there in below.**

code writhed in class file like **myPerson** and below writhed div tag,

<div \*ngFor=”let **X** of myPerson **| keyvalue** ”>//if you want display Object value must be use **| keyvalue** then only it will work

<h2>{{ **X.key** }} -- {{**X.value**}}</h2>

</div>

Save it and execute.

Comment above div tag and execute below example Ex:2

**EX:2**

**(Interview Question)** how can you to get the values from array of object.

<h3 \*ngFor=”let **X** of **myPerson**”> **//internal loop repet chesi outer loop bind chestu key value property use cheyali**

<h2 \*ngFor=”let **Y**of **X**” **| keyvalue**>

{{**y.key**}} -- {{**y.value**}}

</h2><hr>

</h3>

**ng-for.component.ts**

export class NgForComponent {

public **myName** = [‘smith’, ‘Jhone’, ‘ward’] //give this property name to view file.

Public myPerson = { //displaying Objects

“id” : 1,

“name”: ‘smith’,

“age” : 23

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

(video-11)

**Services**

**Before going to services we discusscing what happen with out Service ? so for that we want two component**

**>ng g c components/empName --skipTests=true (presenter)**

Our intention is display employname

**>ng g c components/empInfo --skipTests=true (presenter)**

Our intention is display Remaning details Like id,cours like that.

Give **empName** selector name to root component as a element **Ex:** <app-emp-name > </app-emp-name>

Give **empInfo** selector name to root component as a element **Ex:** <app-emp-info > </app-emp-info >

**emp.nam.component.html**

<h1>Employee Names</h1>

**emp.info.component.html**

<h1>Employee Information</h1>

Save and execute.

**Ex:2**

**emp.nam.component.ts**

Public myPerson = **[**

{ //displaying Objects

“id” : 1,

“name”: ‘smith’,

“age” : 23

},

{

“id” : 2,

“name”: ‘jhone’,

“age” : 24

},

{

“id” : 3,

“name”: ‘ward’,

“age” : 25

}

**]**

**emp.nam.component.html**

<h1>Employee Names</h1>

<h3 \*ngFor=”let x of **myPerson**”>

<h2>{{x.name}}</h2> //displaying only **employeee names**.

</h3>

**emp.nam.component.css**

h2{

color:red;

}

Save and execute. And we want employee Information so go to employee information component.

**Ex:3**

**HERE**  we are displaying reaming or **all values**

**emp.info.component.html**

<h1>Employee Information</h1>

<h3 \*ngFor=”let x of **myPerson**”>

<h2>{{x.id}} -- {{x.name}} -- {{x.age}}</h2> //displaying **all values**

</h3>

Note: we will get error why because we not declaring **myPerson** in our class now we are declaring.

**emp.info.component.ts**

Public myPerson = **[**

{ //displaying Objects

“id” : 1,

“name”: ‘smith’,

“age” : 23

},

{

“id” : 2,

“name”: ‘jhone’,

“age” : 24

},

{

“id” : 2,

“name”: ‘jhone’,

“age” : 24

}

**]**

**emp.info.component.css**

h2{

color:blue;

}

Save and execute. You will get result but this is not correct processer.

Why because we are providing same pice of data several palaces like **EX:-**  **emp.info.component.ts and emp.nam.component.ts** don’t do like that

we must fallow **Dry Principal** (dry stands for – **do not repeat yourself**)

**Dry Principal**

This principal will says if you have any data or information keep it in one place do not keep multiple places. If you place multiple places you will get Redundancy (it means duplicate) then you will get anomalys ( its mean problems)

Q) what type of different anomaly what we are getting if we are placing same pees of information several places ?

A) we will get 3 types of anomaly

1.insert anomaly

2.update anomaly

3.delete anomaly.

Q)what is insert anomaly?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Services** (video-12)

As of now we don’t have database so we are creating one database like one folder.

Open **assets** 🡪inside of that Creating Foder “**data**” 🡪 inside of that crating file **“empData.json”** there copy data from **emp.info.component.ts** past it in **empData.json** and add sum feelds **// below exaple**

**EX: asstes**

**|--data** // this iswe treated as **DataBase**.

**|--empData.json** // placing data in **json** values **single cots** not allowed use **double** **cotes**.

**[ //**providing value

{

“id” : 1,

“name”: “smith”,

“age” : 23,

“course”:”angular”

},

{

“id” : 2,

“name”: “jhone”,

“age” : 24,

“course”:”angular”

},

{

“id” : 2,

“name”: “jhone”,

“age” : 24,

“course”:”angular”

}

**]**

**Creating service**

>ng g s services/empData --skipTests=true (presenter)

**Note**:- if you create new **service** must be give our service name on root module(**app.module**) in **provider[]** section

**EX:** **app.module**

**providers:** [**EmpDataService**], //Go with suggestion above automatically imported other wise you need to write,

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

For this **EmpDataService** service we are passing request to the server, which type of request we are sending **HTTP** **requests** we are passing it may be **GET**,**POST**,**PUT**,**DELETE**. For this service we want one module that is **HttpClinentModule**

why because **GET**,**POST**,**PUT**,**DELETE** methods available in **HttpClientModule**

so goto the **app.module** in import section write code. See Below example

**EX:**

**app.module**

import { **HttpClientModule** } from ‘@angular/common/http’;

copy above **HttpClientModule** class and past it below **import[]** section.

**Import**[

**HttpClientModule**

]

**emp.data.service.ts //**this is service module, it is available under services folder

**export class EmpDataService {**

**private** **\_URL** **=’../../assets/data/empData.json’** //4th Step:- Constructiong URL, and provide this \_URL as a argument

**constructor(private \_http:HttpClient ) { }** //2nd Step:- injecting HttpClient class as a private property(httpClient instance crating) .//Go with suggestion it is automatically imported. //private property inside of the class only use.

**getEmployee(): observable< empInterface[] >{** // 1st Step:- writing method, by using getEmployee method passing one request, if we want use GET request That GET method there in HttpClient Class //So if we want use that GET method goto the constructor inject **HTTPClass.**

//8th Ofter passing get request receiving **observable** and added type, type is **< empInterface[]>**

**retrun** **this.\_http.get< empInterface[] >(this.\_URL)**

//3rd , get method is accept URL (mean where our data located provide as a argument, in this class only above we will construct one URL)

//(5th **\_URL** providing argument)

//6th Ofter creating interface, add interface class name to get method, and array of object there in data file so add array **empInterface[] you** will get error **must be** **export** from interface then it will work

//7th service is passing the get request and after server processing we will receiving OBSARVABLE (response)

// 9th must be **return** the observable value

**}//method close.**

**}//class close**

In our **data** file one component will taking Only Names and Another component will take all data taking, **So if you want particular data use Interface** that is there in Second class

**Creating one Interface**

**App 🡪** under app take new file **myEmp.ts**

**App**

**|--myEmp.ts** // this is file.

**myEmp.ts**

**export** interface **empInterface** { //this interface is using in service so you must **export** this interface // provide this interface name to service class inside get method.

id: number;

name: string;

age: number;

// we are not taking **course** property why because we don’t want that feeld

}

**emp.nam.component.ts**

export class **EmpNameComponent** implements **OnInit** {

public **myPerson** = []

constructor (private **\_empService**: **EmpDataService** ){ } // 1st step : we want to use **EmpDataService** that isclassname in above, that is **EmpDataService** declare as private attribute

go with suggestions automatically imported.

ngOnInit() { //when the component is opened we want data so use lifecycle hook

this.**\_empService**.**getEmployee()** //2nd use constructor property and you will get method name and go with suggestions automatically imported.

.**subscribe(res** => this.**myPerson** = **res,** //3rd inside of **EmpDataService** class we are returning **getEmployee()** method So we are **subscribing** here // when the

**subscribe** time response(**res**) assning to local property **myPerson. \\** and **response (res)** is coming from **service**

**err** => console.log (“Error while receiving data from service:” **+err**))// sapose when the response come to the service if you face any **error** putting in to the console **my perpose**

}

}

**emp.info.component.ts**

//Same proceser fallowing here mean (emp.name.component.ts)

Export class EmpInfoComponent implement OnInit{

Public myPerson = []

Constructor(private \_empService: **EmpDataService**) { }

ngOnInit() {

this.\_empService.getEmployee()

.subscribe(res =>this.myPerson = res,

err => console.log(“Error while receiving data from service:” +err))

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

(Video-13)

**Service Error Handling (Http Error responses handling)**

In above Server is sending replay

but incase particular URL data is not there means **Emp-data.service.ts 🡪** class inside **Private \_URL = ‘../../assets/data/empData.json’,** not found means what I need to do Add **Pipe**

**EX:-**

**Emp-data.service.ts**

**Import { catchError } from ‘rxjs/operators’;** // **2nd** importing catchError method.

export class EmpDataService {

Private \_URL = ‘../../assets/data/empData.json’ // if this data url is not there or mismatch we need to handel server replay so add pipe

constructor(private \_http: HttpClient) { }

**getEmployee**(): Observable<empInterface[]>{

return this.\_http.get<empInterface[]>(this.\_URL)

.**pipe**(**catchError(this.myError)**) //**1st** catching error from server so using **pipe** // **catchError()** method there in **rxjs** packages so go above and import it,//

//**3rd** passing argument **this.myError**

}

**myError**( **error** : **HttpErrorResponse** ){ //4th defining **myError** method and server sending **error** response that is there in **HttpErrorResponse** go with suggestions automatically imported.

**return** **throwError** (**error**.**message**) //5th we must **throw** the error to component

//6th what error we need to send, here **HttpErrorResponse** we are created instance that is **error** inside of **message** we are sending

//7th we are throwing so we **return** the **error**.

}// myError colose

}

Note:-  **suppose any error come from service I what to show to view file so lets take component class file**

**emp-name.component.ts**

export class EmpNameComponent implements OnInit{

public myPerson =[]

public **myError** = ‘ ‘ // **2nd** supposeany error come from service I what to show to view file, displaying empty string.

constructor(private \_empService:EmpDataService){}

ngOnInit() {

this.\_empService.getEmployee()

.subscribe(res =>this.myPerson = res,

err => **{** //**1st**  here we are printing error console so we are taking group of statements using **{ }**

this.**myError** = **err**, //2nd service throwing **error** I am assigning to local property.// and i want to show class property so open view file by using interpolation {{ }}.

console.log("Error while receiving data from service:" +err) **}** )

}

}

**emp-name.component.html**

<h2>Employee Names</h2>

<h2 \*ngFor ="let x of myPerson">

<h3>{{x.name}}</h3>

</h2>

<h3> {{ **myError** }}</h3> //**1st**  in employee Name components with out data if any error comes by using interpolation we will see what is the error.

Save it execute .

it will work property and go above service file (**Emp-data.service.ts )** file do sum url chages then you will find error this is URL I am adding number **2** to miss match the data EXL:- Private \_URL = ‘../../assets/data/empData**2**.json’

Note:- previously we are discussing **data binding** there with in component only communicateing **class files** to **view files** and **view file** to **class file**.

**Component Communicating**

Q) How can you communicate between the components ?

A) we have several ways

1st Poing :by using input output decorators we will communicate with components.

2nd Point : and by using Services also we will communicate with components.

myUnder standing purpose **Ex:-** empNameComponent and empInfoComponent both are using Service

3rd Point : by using behavioral subject we will communicate.

4th Point: by using stores we will communicate.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

(video-14) (imp)

**Two different component between communication**

**Note: Here by** using input and output decorators we are sending information parent component to child component and child component to Parent component we are discussing,

In this example we are taking **app.component.ta** as **parent component**

**Ex:1** **Parnent component** **TO** **child component communicating**

We are sending **parent** **to** **child** so we are creating child component

>ng g c components/child --skipTest=ture (presenter)

Give **child** selector name to root component as a element **Ex:** <app-child > </app-child >

**app.component.ts (parent)**

export class AppComponent {

public parentMsg = “This is Parent Message” // 1st declared one property assigned one value. // as of now we are taking static value in real time they will take dynamically .

// 2nd parentMsg value I what to send **child component**. How it is work means? we are binding parent property file in view file.

}

**app.component.html (parent)**

<app-child **[fromParent] = “parentMsg”** ></app-child> //3rd property binding with parent class property. EX:- **[fromParent] = “parentMsg”** //I want to send this property file to child.

**child.component.ts**

export class childComponent **{**

**@Input()** public **formParent**; //4th parent component property we binded as **fromParent** name that binded name we are taking chaild class property.

//5th So it will tread as normal property so give decorator before variable that is **@Input()**

**}**

**//** I want to print chaild class property values in view file

**child.component.html**

<h2>child component: {{ **formParent** }} </h2> //by taking child class property and using interpolation we are printing parent value. // with out confusion we are writhed component name

**EX:2 Child component TO Parent component communicating**

Q) child component to parent component we can not send message directly why because?

A) child component inside parent component selector is not there that’s why we can not call directly .

//in child component any event by doing **emit** we will send data from child to parent

**child.component.html**

<h2>child component: {{ formParent }} </h2>

<button type=”button” class=”btn btn-success” (click)=”**onClick()”>**Click me</button>// 1st taking one button racing event that is **(click)** and Event handler that is **“onClick()”**we are defining in child class

**child.component.ts**

export class childComponent **{**

**@Input()** public **formParent**;

**@Output()** public childInfo = **new EventEmitter()**  **//3nd** crate public childInfo

**//5th** and create instance for **EventEmitter()**

**//6th** emiting means going to out so must be declare **@output() //** after send this value to parent component so go there.

**onClick(){ // 2nd** child view file event handler defining here. And inside of handler add **emit** property.

this.childInfo.emit(“this is child Response”) **//4th** we are taking childInfo and emit the inside of message **EX:** this.childInfo.emit(“”)

**}**

**}**

**app.component.html**

<app-child (**childInfo**)= **“message = $event”** ></app-child> //**7th** by using event we are **emit** by taking **childInfo** property so take **childInfo** property do event bind.// if you **emit** value in angular we have special key it will bring the values that keyword is **“$event”** and you should be assign local property thais **message //** we are declaring message as local property in parent class file

<h2>parent component : {{message}} </h2> //9th displaying message by taking parent class file.

**app.component.ts**

export class AppComponent {

public parentMsg = “This is Parent Message”

public **message** = ‘ ‘; // **8th** we are declaring as empty taken from view file. // in parent component view file we are displaying.

}

Save and execute, if you press the click me key then only child file going to parent file.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

(Video-15)

**Router**

In angular if we create component that component selector name must be give in root component view file then only it will work

Creating component for routing.

>ng g c components/home --skipTests=ture (presenter)

[**Home.component.html**](http://Home.component.html)

<h2>Welcome to Home Page</h2>

[**Home.component.css**](http://Home.component.css)

H2{

Text-aline : center;

Color: blue;

}

**forExample:**

**app.component.html**

**<** ! **- -**  **<app-home></app-home> - - > //**commenting his line and using **Routers**

**<router-outlet></router-outlet> //5th** router outlet declaring in root component. Save and execute it.

So we can not give all components names in root component the list will huge that’s why Use Routers.

Open router folder

**app-routing.module.ts**

const routes: Routes = [

**{** //**1st** every component router treated as one Object by using curly brackets **{ }**

//**2nd**  before write path please check **index.html** inside **base** tag **href=” / ”** (must be forward slash)

**path : ‘ ‘ ,** // **3rd** if path is empty url after the localhost:4200 **redirect** to home page. And // don’t give double space. It will not work.

**redirectTO : ‘/home’** **//3.1** url constricting home component. EX: localhost:4200/home

**pathMatch: ‘full’** **//3.2**  if the url is full executing component.

**},**

**{**

**Path : ‘home’,** //**3.3** after localhost:4200 home there means Open HomeComponent.

**Component: HomeComponent** //**3.4** Opening HomeComponent Go with suggestions.

**},**

**{**

**path: ‘empName’ // 6**th service we gave rong url change it it will work,

**component: EmpNameComponent**

**},**

**{**

**path:’empInfo’, //7th step.**

**component: EmpInfoComponent**

**},**

**{**

**path:’ \*\* ‘, //10th** pagenotFound component configured as two \*\* this is we called **whild entry** must be write in last Router if we write any component before control will not go down. Save and execute it. **11th step is below.**

**component: PageNotFoundComponent**

**}**

**//8th** for sapose we are searched unknown component name in url then that time we want page not found component. So creating component.

>ng g c components/pageNotfound --skipTests=true (presenter)

**pageNotFound.component.html**

**<h2>Page Not Found 404 Error </h2> //9th** after this step configure this component in router

];

//**4th** Save it and execute it will not work, why because must be give **an root component** **view file inside** **<router-outlet></router-outlet>** then only it will work. **Goto root component view file**.

**//11th** we know how to search but user don’t no how search that’s why we are creating buttons by clicking the button they will get information by using (click) event.

**app.component.html**

<! -- <button mat-raised-button color=”primary” **routerLink=”/empName”**>Emp Names</button> -->

<button mat-raised-button color=”primary” **[routerLink]=” ‘/empName’ “**>Emp Names</button> /**/12th** by using **routerLink**  if I press the button empName component will Opned. And give correct **path name** we writed in router module that is EX:- inside of **app.routing.module.ts** give this path name path: **empName**, // routerLink configuration name must be empName Component name that we decalared in router module path name

/**/13th** We can use both option **routerLink=”/empName”** or **[routerLink]=” ‘/empName’ “** angular team provided two options

**//14th** we can use any one but sir used property binding that is **[routerLink]=” ‘/empName’ “ ,**

<button mat-raised-button color=”primary” **[routerLink]=” ‘/emeInfo’ “**>Emp Names</button> **// 15th** another button for empInfo

**Note**:- previous le we are not added material so created material module and added button component form angularmaterial website.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**(video-16)**

From to day onwards new application developing .

**For database purpose 🡪 mangoDB.**

**For Server purpose 🡪 Express Server.**

**BackEnd purpose 🡪 Nodejs.**

**FrontEnd purpose 🡪 Angular.**

So First We are doing server setup.

**ServerCreating**

Simple and efficient code we are creating server.

So goto up there select application name for Example: **HeaderApp** in our application and another Example is Project name is : **AngularApp** there select it and create folder. And forlder name is **server //** I creatd server folder under **HeaderApp** folder but, it is created in **out of** **src** folder that’s why I created **under app** then it is created Server folder.

We are just created folder name is server, but not server, so that folder want to behave like a server, so do the initialization of the server.

**Q)** How to initialization the server ?

A) goto the server folder in terminal Right click and select **open in a integrated Terminal** and there wire command writed in below see there

>**npm init --yes** (press) then we will get one **package.json** file is created.

//Express server installing same server folder

**Note:** We are install express server ForntEnd angular application handling purpose express server wants support that is middleware that middleware name is **body-parser.**

And same terminal installing **express server** and **middleware. body-parser**.

**>npm install express body-parser - -save (presenter)**

Under server folder we are created only **server.js** folder

**Server 🡪 server.js**

**Servers.js //**installed **express server** **require here**

const **express** = require(‘express’) //1 requiring express server //take any property name but **express** is relevant name that’s why we taken express name.

const **bodyParser** = require(‘body-parser’) //2 body-parser also requiring.

//3 by default our application is ruing on PORT Number like server also depends on PORT Number

const PORT = 3000 //4 server port number defining.

const **api** = require(‘./routes/api’) //( ./ means one step up ) requiring file from api.json **( after api.json 1st step)**

//So **body-parser** want use means how we used by taking help of server instance property that is **app** similarly if I want use above **api** take help from **express() server** property that is **app**. And write code below **body-parser ( after api.json 2nd step)**

Const **app** = express() //5 express server instance creating

**app**.use(**bodyParser.**json()) //6 by using server instance property only body parser use.// witch format we using 🡪 we are using **json()** formant

**app.use**(**‘/api’**,**api**) //using **server property** and **‘/api’** you can give any name //and taking above require property **api. ( after api.json 3rd step)** save and execute. How to check means Goto the browser and search **localhost:3000/api** (presenter) you will get api response

**Note**:- if you did any server updates must be stop the server and start the server. EX: stop 🡪 ctrl + C and start 🡪 node server

**we did correct installation or not checking purpose only below code using . By sending one GET request to the server**

**app**.**get**(‘/’, **(req**, **res)** => **{**  //by using **app** sending //**get** Empty rout sending //fat arrow function() //passing **req**, and **res** response coming from server //by using fat arrow method. **=>{ }**

**res** .**send**(“this message from SERVER”) //**res** response coming from server.// server **send()** message

**})**

We want response message means first run the server so that server depends on port number

**app.listen(PORT, () =>{** **//** response coming from server we have to listen that response, //where it will listen 🡪 we have defined port number // unnamas function writed () // fat arrow method **=>{ }**

**console.log(“Server is running PORT No:” +PORT)** //displaying on console ()

**}) //**save it and execute.

**How to start server (starting server)**

In terminal > **node server** (presenter) // in terminal you will find **output** is -> Server is running PORT No: 3000

And if you check port number 3000 in a new browser you will get Output from browser is -> **this message from SERVER**, search like this Ex: localhost:3000

**Note:**

By using our application passing request to the SERVER and every request is **(API request)** //and if we are passing request means Sending routs only.

When the application complicity is HI than you will get confuse and you will put all routes in single folder DeBug time it will simple to use.

Under server folder we created one folder that is routes under router folder created file file name id api.js

Server 🡪router (forlder) 🡪 api.js (file)

**api.js**

// require server here

**const express = require(‘express’)** //require server here

why we created api.js file to handle routes

**const router = express.Router()** // express server router packages ki instance created

//so lets check by router any message will come from server or not by using get requst.

**router.get(‘/’, (req, res) => { //**assume request passing and assume response coming from server. By using fat arrow method

**res.send(“This is from API Router”)**  //response coming from server.

**})**

if you want use another location must be use **export**

**module.exports = router //**router taking from above instance property **.**

**Note**:- api.json file required to server so Goto above **server.js** folder write PORT number below.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

(Video-17)

**Database Setup** (watch database stup) on Angular full course.

In visual studio also we can setup database so I am going direcly.

Open Browser search this link: **account.mangodb.com/account/login** copy this link and past it in browser.

Login with your gmail

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

From Angular version 10 rsk YouTube Chanel video

**PIPES**

Q) What is mean by pipes

a) Pipes are use to transforming incoming stream of data into required format before displaying view

that means sapose

sum data is coming into view to display it so before displaying we are going to transform the data into our required format

inTelugu 🡪 (So mana dagara data vunapudu screan meda ante html view file lo display cheyaka mundu manadagara vuna data ni kavalasina format lo convert chesi display chestamu.)

**We have different types lets try**

**test.componnt.ts**

public date = new Date(); //use this date property in view file

public name= “Smith”; //we declared several properties

public msg = “Welcome to angular”;

public person = {

“firstName” : “SmithKumar”,

“country” : “United states of America”

}

**test.componnt.html**

<h2> {{name}} </h2> // using name property by interpolation.

<h2> {{name **|** lowercase}}</h2> // **by using pipe symbol converting all name value to lower case**.

<h2> {{name **|** **uppercase** }}</h2> // all are upper case.

<h2> {{msg **|** titlecase }}</h2> // msg property word 1st value is Capital **EX**: **W**elcome **T**o **A**ngular

<h2> {{name | slice: 2 }} </h2> // in name property output start with ofter 2 index start with 0 outPut is:- ith

<h2> {{name | slice: 2: 8}} // start with 2 and last before 8

<h2> {{ person | json}} </h2> // object property converting json format

// number pipes Examples

<h2> {{5.678 | number: ‘1.2-3’}} </h2>

<h2> {{5.678 | number: ‘3.4-5’}} </h2>

<h2> {{5.678 | number: ‘3.1-2’}} </h2>

//percent Examples

<h2> {{0.25 | percent}} </h2>

// currency Operator

<h2> {{0.25 | currency : ‘INR’ }} </h2>

//date Example

<h2>{{date}}</h2> //create date instance in class file Go and check it.

<h2> {{date | date : ‘short’ }}</h2> // by using pipe displaying date.

<h2> {{date | date : ‘shortDate’ }}</h2>

<h2> {{date | date : ‘shortTime’ }}</h2>

<h2> {{date | date : ‘medium’ }}</h2>

<h2> {{date | date : ‘mediumDate’ }}</h2>

<h2> {{date | date : ‘mediumTime’ }}</h2>

<h2> {{date | date : ‘long’ }}</h2>

<h2> {{date | date : ‘longDate’ }}</h2>

<h2> {{date | date : ‘longTime’ }}</h2>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

From Angular version 10 rsk YouTube Chanel video

**Dependency Injection (watch again)**

1.without Dependency Injection

2. without Dependency Injection as Design Pattern

3. without Dependency Injection as Framework (Provided by Angular)

Note: - instance means Object Creating we created employe Ex: public employee = new employee()

Angular dependent take as Desinepattern

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Forms:** (video-30)on angular-10 youtub

* Forms are vary important for any business application.
* Forms are used to create an experience that guides the user efficiently and effectively through the workflow.
* As a developer we have do,

\* Data Binding

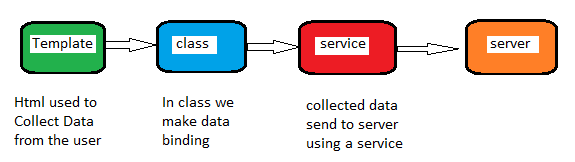
\* Change Tracking

\* Validation

\* Visual Feedback

\* Displaying Error Messages

\* Form Submission

****

Above flow is by using html template collecting data from user, and collected binding in a class, by using service collected data sending to the server

So in this way we to do we have two type of forms

**Two Approaches:**

1. **Template Driven Form** – Most of the code written in html file.
2. **Reactive Forms / Model Driven Forms** – Most of the code written in Component Class.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

(video-31)on angular-10 youtub

**Template Driven Forms**

* Template driven forms are forms where we write logic, validations, controls etc, in the template part of the code (html code).
* The template is responsible for setting up the form, the validation, control, group etc.
* Template driven forms are suitable for simple scenarios, uses two way data binding using the [(NgModel)] syntax.
* But unit testing might be a challenge.

**How to enable Template driven forms**

* To use the template driven form, we need to explicitly and import on **app.module.ts**  and declare **FormModule** in **declaration[] section EX:**  **import {FromModule} from ‘@angular/forms’**.

**ngModel:**

we need the ngModel in the form input and the input must be named

**EX:**

<input type=”text” ngModel name=”firstName”>

* There are cases where we need pass an event listener to the input field, or pass the value of the input to out component, we need assign a template reference variable to the input

EX:

<input type = “text” ngModel

Name=”firstName”

**#firstName** = “ngModel” // by using #

(any-event)=”event-Handler(**firstName**)”>

// if you want write one input field particular event lizener so that particular element use Reference variable and # represent Reference variable.

**ngForm:** (if you want use GroupOf elements use ngForm)

the ngForm is an instance of the FormGroup. The FormGroup represents the group of FormControl, each form is a FormGroup because it will have at least ne FormControl that given us access to (ngSubmit) which can be bind to a method in our component.

**Ex:**

//every elements is a control like Button, checkbox , inputbox etc. those are form controls.

// So we puted is inside **form tag** those are called in FormGroup

// inside of **form** **tag** we want use **all elements** like buttons and inputbox etx. if you want use all form use **ngForm** and as well **(ngSubmit)** then we can access all controls values inside of **Form.**

<form **#formName** = “**ngForm**” **(ngSubmit)** = “submit(**formName**)”>

<input type=”text” ngModel

Name=”firstName”

#firstName =”ngModel”

(any-event) = “eventHandler(firstName)”>

</form>

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Peding vidos

(Video-17)

**Database Setup** (watch database stup) on Angular full course