Lets start…

Sections

Topic

points

**1:Getting started**

**Create new application ->**

Ng new myapp --no-strict --standalone false --routing false

**Run**

Ng serve

**Installing packages**

Npm I --save bootstrap@3

@stands for version

**Create new component->**

ng generate component xyz **or** ng g c xyz

**what is ngmodule and formsmodule?**

**Ngmodule**- It is a latest angular (not angular.js) modularization mechanis m which is used for organizing our application into block of modules.Module is nothing but the encapsulation of components and services.

**Formsmodule-** provides directives and services for working with the forms.it enables two way data binding using “ngModel”.

**Typescript**

Strong type checking

**2:The basics**

**Component**

**template property of the component**

Component should have **template** attribute (to define the template in the component.ts file itself) or **templateurl** attribute (to give the link of component.html file to be used as the template) in its .ts file

Similarlly, css can be written in css file or component.ts file using the attribute **styles** as styles:[`h3{color:red}`]

**Selector property of the component**

**Selector : ‘app-server’ ---->**

<app-server></app-server>

**Selector : ’ [app-server] ‘ ---->(attribute selector)-->**

<div app-server></div >

**Selector : ’.app-server’--->(css class selector)--->**

<div class=”app-server”></app-server>

**Data binding**

**1.send data from .ts to .html**

**String interpolation** {{data}}

**Property binding** [property]=”data”

**2.send data from html to .ts**

**Event binding** (event)=”expression”

**3.two way data binding (combination of 1 and 2)**

[(ngModel)]=”empname”

**String interpolation**

It is used to insert something in html template.

Accepts all kind of expressions which returns the string or which can be easily converted to a string (e.g. number) .

We cant use multiline statements ( using if else ) in this expression.

We can use ternary operator as an expression.

E.g. {{servername}}

**Property binding**

It is used to change the value of html attributes of the html tags.

Every html element is parsed by the browser and added to the DOM .All those DOM elements have multiple properties which can be changed by angular at runtime using property binding.

E.g. [disabled]=”truefalseflag”

**Event binding**

Used to perform operations on event (e.g onclick)

e.g. (click)=”executefunc()”

**Two way data binding using ngModel**

ngModel which is used with input tag is a directive from formsmodule which is used for two way data binding i.e. we can change the values of a field from input box(frontend/template) as well as from the class itself(backend) .

ngModel- combination of event binding and property binding

e.g. [(ngModel)]=”empname"

**Directives**

Directives are the instructions to the dom.

**structural directive**

e.g. **\*ngIf** - \* indicates that it is a structural directives which means it will change the structure of dom

e.g. <p \*ngIf=”flag”>username <p>

if flag is true then only the username will be shown in the template

we can use else part in the same as follows

<p \*ngIf=”flag; else anothertext”>ashish</p>

<**ng-template** #anothertext>

show this text when if condition fails

</**ng-template**>

**\*ngFor-** it is a structural directive

E.g. \*ngFor =”let i of arr”

We can also use indexing as follows

1. g. \*ngFor=”let I of arr;**let i=index**”

Here we can use index of the current iteration as well

**Attribute directive**

**[ngStyle] –**change the styling dynamically

1. g. [ngStyle]=’{backgroundColor:getColorfunc()}’

**[ngClass] –**add or remove classes dynamically

1. g. [ngClass]=”{‘btn btn-danger’:flag,’btn btn-success’:!flag}”

Here the classes will be added according to the flag

**Structural directives are used to add or remove the elements whereas attribute directives are used to change the element they were placed on .**

**5. Components and databinding deep dive**

**@Input() -**

By default all the component fields (class data memebers) are private and cant be accessed outside the component.So to make them accessible outside the component then we should use **@input()**

So any component which is using our component using its selector will be able to access the components fields.

**VIDEO 71**

**E.g. @Input() name:string;**

**E.g.** @Input(**username**) name:string - Alias to the component field which can be used outside the component.

**@Output()-to inform parent about changes in child**

To make the childs event accessible or listenable from the parent element using **@output.**

**E.g. in child - @Output() myEvent =new eventEmitter<{‘name’:string,’age’:number}>();**

**E.g. in child - @Output(‘anothername’) myEvent =new eventEmitter<{‘name’:string,’age’:number}>();**

Anothername -- is an alias here which can be used whenever refering to this event

**View encapsulation-**

By default (in html web devp world) css written in any file/component is applied for the whole document irrespective of in which component it is written .

But in angular css written in the file /component is only applied to that component ifself.This feature is implemented in angular by assigning and unique selector to all the elements of the component.

If you inspect the project and go to elements section then you can see that -- there is an unique selector for the component and all the elements mounted by that component will have that same unique selector.

This unique selector changes for every component so that angular can uniquely identify the which elements belongs to which component.

**E.g.** \_ngcontent-ng-c406225580

If you want that, css written in any file should get applied to all the document then you should use line below in .ts file as key value pair of @component decorator

**Encapsulation:viewEncapsulation.None**

**VIDEO76**

**Using local reference in templates -**

We can use local reference to all the html elements using **#**

**<input type =”text” #myElem></input>**

We can directly use this keyword in html template

As (onclick)=”**func**(myElem);

In **.ts file**

**Func**(inputelem:HTMLInputElement){

Console.log(inputelem.value);

}

**-------------------Or ----------------------**

We can use this keyword in .ts file using **@viewChild**

**Eg. In .ts file**

**@ViewChild(‘myElem’,{static:false})myinputelem:ElementRef;**

And to get its value we can use

**Myinputelem.nativeElement.value.**

**VIDEO79**

**Using <ng-content></ng-content> -**

By default any content written in between the angular component tag is neglected by angular

E.g <mycomp>**<p>this is paragraph</p>**</mycomp>

This <p>tag will be neglected

If you want this paragraph to be visible inside the mycomp.html then add **<ng-content>** in mycomp.html where you want this para to show

**VIDEO80**

**Understanding content lifecycle -**

**Hooks-**

**1.ngOnChanges -**

Gets invoked Whenever properties decorated with @input() changes .

**2.ngOnInit -**

Gets invoked Whenever component is initialized.

**3.ngDoCheck -**

Gets invoked Whenever change is detected.

**4.ngAfterContentInit -**

Gets invoked Whenever content (ng-content) has been projectedd into view .

**5.ngAfterContentChecked-**

Gets invoked Whenever projected content (ng-content) has been checked .

**6.ngAfterViewInit-**

Gets invoked Whenever components views and child views has been initialized.

**7.ngAfterViewChecked-**

Gets invoked Whenever components views and child views has been initialized.

1. **ngOnDestroy -**

Gets called when component is about to destroy