**Data binding:**

{{}} string interpolation

// call a method in side {{}}

**Property Binding**

[disabled]=”!properryName”

EventBinding

(click)=”func()”

(click)=”func($event)”

$event :Event

(<HtmlInputElement>event.target).value

//two way databinding

[(NgModel)]

**Directives:**

//

\*ngIf=”test; else noserver”

<ng-template #noserver></ng-template>

[ngStyle]=”{‘background-color’:red}” or [ngStyle]=”{backGroundColor:red}”

Or [ngStyle]=”{backGroundColor:getcolor()}”

.online {

Color:red;

}

[ngClass]=”{online:sever===’online’}”

\*ngFor=”let mama of servers; let I is index”

**Communication between component:**

<cockpit>

<serverElement>

@Input() element:{type:”ff”,name:”rojer”}; before single serverelement in <serverElement> so proud

[element]=”serele”

Alias name

@Input(‘praba’) element:{type:”ff”,name:”rojer”}; before single serverelement in <serverElement> so proud

[praba]=”serele”

eventEmitter for clidk event “@angular/core”

**View Encapsulation**

To apply local component css rule across the app

@Component({

Encapsulation: ViewEncapsulation.None;

})

**Local reference:**

**///! Only in the view template not in type script file**

<input #serverNameInput/>

<div (click)=”callfun(serverNameInput)”

Calfun(dfdf:**HTMLInputElement**) /// type

**@viewChild()**

@viewChild(‘serverNameInput’) severelemnt:ElementRef;

Serverelemnt.nativeElement.value // access value

**<ng-content> // place between the content tag**

**Component life cycle:**

**ngOnChanges // called after a bound input pproperty changes**

**ngOninit // once the component is initialiezed after constructor**

**ngDoCheck // every change detection run**

**ngAfterContentChecked // every time the projected content has been checked**

**ngAfterViewInit // after component’s view (and child views) has been initialized**

**ngAfterViewChecked //every time the view and child view is checked**

**ngOnDesroy //**

**ngContentChild()**

**Directive**

Custome directive

@Directive({

Selector:”[highlite]”

})

Export class directive{

Constructior(private eleref:ElementRef)

}

ngOnInit(){

this.eleref.nativeelement.style.background.color=”red”;

}

! include in declearation

Cli : ng g d –spec false

///better Directive

@Directive({

Selector:”[highlite]”

})

Export class directive{

Constructior(private eleref:ElementRef ,private render:Renderer2)

}

ngOnInit(){

this. Render.setStyle(this.elRef.nativeElement,’background-color’,’blue’);

}

**//HostListener**

@HostListener(‘mouseenter’) mouseover(eventData:Event){

}

@HostBinding(‘style.backgroundColor’) backgroudcolor:string;

This. Backgroudcolor=”blue”;

//dynamically

@Input() backgroundColor:string;

[backgroundcolor]=” ’yellow’ ”

**Custome StructuralDirective**

@Directive({

Selector:”[appunless]”

})

Export class directive{

@Unput() **set unless(condition: boolean)**{

If(!condition){

This.vcRef.CreateEmbeddedView(this.templateRef);

}else{

This.vcRef.clear();

}

Constructior(private eleref:TemplateRef<any> ,private vcRef:ViewContainerRef)

}

ngOnInit(){

this. Render.setStyle(this.elRef.nativeElement,’background-color’,’blue’);

}

**ngSwitch**

<div [ngSwitch]=”value”>

<p \*ngSwitchCase=”5”></p>

<p \*ngSwitchCase=”5”></p>

<p \*ngSwitchCase=”5”></p>

<p \*ngSwitchDefault></p>

</div>

**Services**

**Service**

**@injectable**

**Routing:**

Import {Routes} from ‘@angular/router’;

appRoutes:Routes =[

{ path:’’,component:HomeComponent}

]

Module.ts imports:[RouterModule.forRoot(appRoutes)]

View: <routerooutlet></router-outlet>

**Navigation with Router:**

routerLink=”’link’” or [routerLink]=”[‘’,’’,5]”

**Attach a css Class for Active Link**

routerLinkActive = “active”

[routerLinkActiveOptions]=”{exact:true}”

**Navigating Programatically:**

Constructor(Private router:Router){}

This.router.navigate([“USER”]);

**Relative to path**

This.router.navigate([“USER”],{relativeTo:this.route});

**Parameter**

{path:’users/:id’}

Constructor(private route:ActivatedRoute){}

This.route.snapshot.params[:id]

This.route.params.subscribe((params:Params)=>{

This.params

});

**Query Params**

[queryParams]=”{allowEdit:’1’}”

[fragment]=”loading”

**Programmatically**

This.router.navigate([‘/servers’,id,’edit’],{pueryParams:{allowEdit:’true’},fragment:’loading’}]

**Child route:**

appRoutes:Routes =[

{ path:’’,component:HomeComponent,children:[

{},{}

]}

]

<router-outlet>

**Preserve queryParams**

This.router.navigate(,,queryParamsHandling:’preserve’)

**Redirects**

{path:’’,component:’’,redirect:’’}

Path:’\*\*’ /// catch all route which is unmatched

**AuthGaurd**

Implement CanActivate{

CanActivate(route: activatedRouteSnaphot,state:routerStateSnapshot):obsebavle<Boolean> |

Promise<Boolean> | Boolean{

.isauth.then(dfef:bool)=>{

Return true;

}

}

// login service /Promise

Const promise:promise = new promise((resolve,reject)=>{

Resolve(true);

});

Return promise;

Path,’’,canActivate:[authGaurd]

**CanActivateChild(){**

**}**