**Over View Of Angular Lessons:**

**Components:**

a) Components Creation

b) Components Interaction

**Binding:**

a) Property Binding - 2

b) Class Binding - 5

c) Style Binding - 4

d) Event Binding – 1

e) Template Reference - 1

f) Two Way Binding - 1

**Directives:**

a) ngIf - 2

b) ngFor - 1

c) ngModel – 1 [( )]

d) ngSwitch - 1

e) Pipes

**Services:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* XXX \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Important Commands:**

node -v >> node version;

npm install -g @angular/cli >> install angular; (npm install -g @angular/cli@latest)

ng --version >> angular version;

code . >> opens MS code;

ng new projName ($secs) >> creates new project;

ng new projName –routing >> create new project with routing;

ng serve ($secs) >> run in browser (http://localhost:4200/);

ng serve -o >> run and opens browser;

ng g c componentName >> Create new component;

ng g s serviceName >> Create new Service;

npm install –save cors; npm install jsonwebtoken --save

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* XXX \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Routing Play:**

main.ts >> app.module >> app.component >> index.html

**Component Functionality:**

**Ex:**

import { Component, OnInit } from '@angular/core';

@Component({

selector: 'app-vinnu',

templateUrl: './vinnu.component.html',

styleUrls: ['./vinnu.component.css']

})

export class VinnuComponent implements OnInit {

public name = 'SATVIK PNNAMGALLI';

ngOnInit(): void {

}

}

**Analysis:**

The selector will fetch the data from templateUrl/template and represents it's instance. This selector instance should exist in other component.html;

The class props can use in below (3) types.

As Class: \*[Selector >> in Component];

selector: '.app- && <div class="app-vinnu"></div>;

selector: 'app-vinnu' && <app-vinnu></app-vinnu>;

selector: '[app-vinnu]' && <div app-vinnu></div>;

**Interpolation:**

To avail the class props in the template url through curly braces known as **INTERPOLATION**.

Ex: <h4>{{ name + " " + getName() }}</h4>

**Property Binding (3):**

Declaring/Binding class property as HTML element property is known as Property Binding.

id="{{name}}" (This way boolean like disabled prop will not work)

[id]="name" (Boolean also work in this way)

\*bind-disabled="isDisabled" (Will work)

**Css Class Binding (5):**

Binding class property as HTML element's class property is known as Class Binding.

<h2 class="text-success">Success</h2>

<h2 [class]="fontStyl">Danger</h2>

<h2 class="{{fontStyl}}">Danger</h2>

<h2 [class.text-danger]="hasError">Error Showing</h2>

\*<h2 [ngClass]="multClass">Multi Classes Appling</h2>

public fontStyl = "text-danger";

public hasError = true;

public isSpecial = false;

public multClass = {

"text-success": !this.hasError,

"text-danger": this.hasError,

"text-special": this.isSpecial

}

**Style Binding (4):**

Binding class property as HTML element's style property is known as Style Binding.

<h2 [style.color]="fclr">First Line Statement</h2>

<h2 [style.color]="'blue'">Second Line Statement</h2>

<h2 [style.color]="hasError ? 'red' : 'green' ">Third Line</h2>

\*<h2 [ngStyle]="clsObj">This line for ngStyle directive</h2>

\*Note: Only color props can apply to [style.color ]; Also quotes must apply for every color prop ‘red’.

**Event Binding (2):**

Binding (mouse) events to the DOM properties know as Event bindings.

<button (click)="onClick()">Click Here</button>

<button (click)="onClick2($event)">Click Here</button>

{{fclr}}

calling functions:

onClick() => this.fclr = "Hello World..";

onClick2(event) => console.log(event);

**Template Reference Variables (1):**

<input #myId type="text">

<button (click)="onClick(myId.value)">Click Here</button>

{{fclr}}

onClick(seee){

this.fclr = seee;

console.log("seee"); }

**Two Way Binding (1):**

In app.module.ts >>

1). import { FormsModule } from '@angular/forms';

2). imports: [FormsModule],

3).

<input [(ngModel)]="name" type="text">

{{name}}

public name = "";

**ngIf Directive (3):**

1). <h2 \*ngIf="true">Welcome to test Event Binder</h2>

2). <h2 \*ngIf="showing">Welcome to test Event Binder</h2>

public showing = true;

3). <h2 \*ngIf="showing; then thenBlock else elseBlock">Welcome to test Event Binder</h2>

<ng-template #thenBlock><h2>Showing Block</h2></ng-template>

<ng-template #elseBlock><h2>Hiding Block</h2></ng-template>

Note: ng-template is mandatory for this ngIf; Must not surrounded with one ele (div);

**ngSwitch Directive:**

<div [ngSwitch]="color">

<div \*ngSwitchCase="'red'">You have picked the color of: RED</div>

<div \*ngSwitchCase="'blue'">You have picked the color of: BLUE</div>

<div \*ngSwitchCase="'green'">You have picked the color of: GREEN</div>

</div>

Note: \* ngSwitchCase is mandatory; Must surrounded with one element (div);

**ngFor Directive:**

public colors =["green", "blue", "white", "red"];

<div \*ngFor="let color of colors; index as i">

<h2>{{i+1}} {{color}}</h2>

</div>

(first as f || last as l || odd as o || even as e);

**Component Interaction (Input & Output):**

**Input:**

1) app.component.ts >> public name = "Raghunadh";

<app-test [parentData]="name"></app-test>;

2) test.component.ts >> import { Component, OnInit, Input } from '@angular/core';

@Input() public parentData;

<h2>{{"hello " + parentData}}</h2>

**Output:**

1) test.component.ts >> import { Component, OnInit, Input, Output, EventEmitter } from '@angular/core';

@Output() public childEvent = new EventEmitter();

<button (click)="fire()">Click Here</button>

2) app.component.ts >> <app-test (childEvent)="message=$event" [parent]="name"></app-test>

<h3>{{message}}</h3>

**Pipes:**

<h2>{{person | json}}</h2>

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

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

<h2>{{5.678 | number: '2.1-2'}}</h2>

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

<h2>{{name | lowercase}}</h2>

<h2>{{name | uppercase}}</h2>

<h2>{{name | titlecase}}</h2>

<h2>{{name | slice:6}}</h2>

<h2>{{name | slice:6:9}}</h2>

<h2>{{name | slice:6:-6}}</h2>

<h2>{{0.25 | currency: 'GBP'}}</h2>

<h2>{{0.25 | currency: 'INR'}}</h2>

<h2>{{date | date: 'short'}}</h2>

<h2>{{date | date: 'shortDate'}}</h2>

<h2>{{date | date: 'shortTime'}}</h2>

<h2>{{date | date: 'longTime'}}</h2>

<h2>{{date | date: 'mediumDate'}}</h2>

**Services(19Les):**

1) Define the EmployeeService class

ng g s serviceName >> Create new Service;

2) Register with Injector:

Import the created service in app.module.ts and mention in providers (imported name);

providers: [EmployeeService],

3) Declare as dependency in EmpList and EmpDetail

constructor(public \_employeeService: EmployeeService) { }

ngOnInit(): void { this.employees = this.\_employeeService.getEmployees(); }

**Fetch Data Using HTTP:**

import { HttpClient } from '@angular/common/http';

import { Injectable } from '@angular/core';

import { IEmployee } from './employee';

import { Observable } from 'rxjs';

1) Included HttpClientModule in app.module.ts

import { HttpClientModule } from '@angular/common/http';

@imports: HttpClientModule

2) Creating employee.json file http\_url purpose.

[ {"id": 1, "name": "vignesh", "location":"kphb"}, {"id": 2, "name": "srikanth", "location":"pamarru"}]

3) Creating one IEmployee Interface for Observable return type

export interface IBrother{ id: number; name: string; location: string; }

4) Injecting HttpClient as Dependency

a) import { IEmployee } from './employee';

import { Observable } from 'rxjs';

b) constructor(private http: HttpClient) { }

c) public \_url: string = 'assets/employee.json';

d) getEmployees(): Observable<IEmployee[]> {

return this.http.get<IEmployee[]>(this.\_url);

}

5) Consuming Service:

import { EmployeeService } from '../employee.service';

public employeeDetails = [];

constructor(private \_employeeService: EmployeeService) { }

ngOnInit(): void {

this.\_employeeService.getEmployees()

.subscribe(data => this.employeeDetails = data);

}

**Routing:**

1) Generating app with Routing:

ng new projName --routing

Ensuring the routing:

1. index.html >> <base href='/'>
2. <router-outlet></router-outlet> in comp.html;

c) generates app.routing.module.ts

d) Importing in app.module.ts

2) Generating components

a) ng g c employeelist && ng g c departmentlist'

3) Configure into app.routing.module.ts

a) configuring in app.routing.module.ts

import { DepartmentListComponent } from './department-list/department-list.component';

import { EmployeeListComponent } from './employee-list/employee-list.component';

const routes: Routes = [

{path: 'departments', component: DepartmentListComponent},

{path: 'employees', component: EmployeeListComponent}

];

4) Importing & Delcaring in app.module.ts:

a) import { DepartmentListComponent } from './department-list/department-list.component';

b) declarations: [

AppComponent,

routingComponents

],

5) In app.component.html

<nav>

<a routerLink="/departments">Departments</a>

<a routerLink="/employees">Employees</a>

</nav>

**Wildcard Route and Redirecting Routes:**

Creating PageNotFound component for unlisted pages and configuring under app.routing.module.ts.

import { PageNotFoundComponent } from './page-not-found/page-not-found.component';

const routes: Routes = [

{path: 'departments', component: DepartmentListComponent},

{path: 'employees', component: EmployeeListComponent},

{path: '\*\*', component: PageNotFoundComponent}

];

export const routingComponents = [DepartmentListComponent,EmployeeListComponent,PageNotFoundComponent]

Note: This wildcard routing path should be last else any page would navigate to pagenotfound component.

Redirect Route Types:

Default >> {path: '', component: EmployeeListComponent};

Prefix >> {path: '', redirectTo:'/departments', pathMatch: 'prefix'};

Full >> {path: '', redirectTo:'/departments', pathMatch: 'full'};

**Route Parameters:**

Applying parameter routing for every element and displaying in url to capture id from url and displaying in department-detail;

1) displaying the list in html with id and name;

<ul class="items">

<li (click)="onSelect(department)" \*ngFor="let department of departments">

<span class="badge">{{department.id}}</span>{{department.name}}

</li>

</ul>

2)

a) creating department-detail component and configuring in routing.module.ts;

ng g c department-detail

import { DeparmentDetailComponent } from './deparment-detail/deparment-detail.component';

{path: 'departments/:id', component: DeparmentDetailComponent}

export const routingComponents = [DeparmentDetailComponent]

b) applying id for url: (in html)

<li (click)="onSelect(department)" \*ngFor="let department of departments">

import { Router } from '@angular/router';

constructor(private router: Router) { }

onSelect(department){

this.router.navigate(['/departments', department.id]);

}

c) receiving id and displaying element: (in department-detail)

import { ActivatedRoute } from '@angular/router';

constructor(private route: ActivatedRoute) { }

ngOnInit(): void {

let id = parseInt(this.route.snapshot.paramMap.get('id'));

this.departmentId = id;

}

<h3>you selected department with id of: {{departmentId}}</h3>

**ParamMap Observable:**

<h3>you selected department with id of: {{departmentId}}</h3>

<a (click)="goPrevious()">Previous</a>

<a (click)="goNext()">Next</a>

this.route.paramMap.subscribe((params: ParamMap) => {

let id = parseInt(params.get('id'));

this.departmentId = id;

})

**Optional Route Parameters:**

**Relative Navigation:**

Through Relative Navigation, no need to pass the url (/employee) segment instead we can pass **this.route** for router.navigation.

This.router.navigate([employee.id], {relativeTo: this.route});

Interview Questions:

this.auth.registerUser(this.registerUserData).subscribe( res => cl(res), err => cl(err))

ng g c admin-home -it -is -skipTests=true