ANGULAR 8

JS framework to create a reactive single page application

SPA: Partial refresh of the page rather than complete page refresh

AngularJs 1.X -2010

Controller

$scopes

-no mobiles devices

-js

Angular 2 >above -2016

Angular 4,5 – 2017

Angular -2018

Angular 7, 8 -2019

**Component**

**-resused**

**-unit testing**

**-multiple languages**

**Typescript, pure script, ECAM script, dart….**

**-Mobile devices**

**-5 times faster**

**Angular 8**

**-Ivy – internal rendering engine – 70 time faster(size is reduced)**

**-Bazel – build the app**

**-Differentail loading – CLI create multiple bundle for browsers**

**NPM- Node Package Manager**

**It has all dependencies**

**npm install -g @angular/cli**

**Path;**

C:\Program Files\nodejs;

C:\Users\31410\AppData\Roaming\npm

**Create a new Project:**

**ng new Angular8Pro**

**ng serve**

**ng generate component Employee (ng g c Employee)**

**Default component – app component**

**Component :**

**It is a class with template and decorators**

**Module:**

**To organize the app into blocks**

**User Module**

**Create user -comp**

**Edit user -comp**

**Update user -comp**

**Admin Module**

**Add Roles- compon**

**Edit roles-compo**

**Manager roles -compo**

**Browser Module:**

* **Support the browser**
* **Get the standard func**

**TypeScript**

* **Superscript to Java script**
* **OOPs – class, object, inheritance, interfaces….**

**Interpolation – {{}} ,print**

**Directive – ngFor**

**Data binding:**

**-Communication between component and DOM element**

**Component.html – component.ts**

**Different types;**

* **String Interpolation – one way data binding**
  + **{{}}**
* **Property binding – One way data binding**
  + **[]**
* **Event binding – two way databinding = property+event**
  + **()**
* **One way data binding**
  + **If data flows from “ts to html”**
* **Two way data binding**
  + **If data flows from “ts to html and html to ts”**

**[(ngModel)] -two way data binding**

**FORMS**

1. **Template driven**
2. **Reactive Form**
3. **Template Driven Form**

* **ngForm – name a form**
* **ngSubmit – event, to submit the form**
* **ngModel – 2way data binding**

**Validation of form Control:**

**Touched -false**

**Untouched -true**

**Dirty**

**Pristine**

**Valid**

**Invalid**

1. **Reactive forms**

* **Unit testing is easier**
* **More code on component, less on html**
* **No data binding is required**
* **formGroup – name a form, html**
* **formControlName – name a control, html**
* **FormControl – group the controls, component**

**Routing and Navigation:**

**Navigation between different views, when user perform certain action**

**Mapping to the component -path**

1. **Configure the routes**
2. **Add link/Programatic navigation**
3. **Router outlet**

* **RouterModule – Module.ts [ bunch of service and directives]**
* **Routes - path**

**{path:””,component:component name}**

* **routerLink, router – navigate()**
* **router outlet - displays the corresponding component when it is called**

**Programmatic navigation:**

**Router – navigate() =[service class]**

* **dependency injection is performed to create the instance**
* **inside the constructor**

**WILD CARD ROUTES:**

**Invalid routes – navigates when the path is not matched**

**Wild card routes should be in the last of the configuration**

**Routes Param:**

**To pass parameter value along with the url**

**ActivatedRoutes – service**

**Grab value:**

1. **Snapshot way**
2. **Observable**

**SERVICE:**

**ng generate service servicename(ng g s servicename)**

* **Share the data across the component**
* **Simple application logics, used in component**
* **External resources, db**
* **Called through DI(dependency injection)**

**Dependency Injection**

* **Instance of the service class**
* **Constructor of the specific component**

**HTTP SERVICE**

**HttpClient**

* **Service class, http methods -get,post,put,delete**
* **Perform http request**
* **Each req has different signatures and returns values based on different input**
* **RXJS libraries – Observable**

**Observable:**

* **Handles multiple values**
* **Pass msg between different parts of the application**
* **Value that the body return will be the Observable of <T>**
* **Asynchronous pattern**
* **Lazy**
* **They are cancellable**
* **In this pattern, we have Obervable ad Observer**
* **Observable, emit the data on notification**
* **Observer also called as subscriber**
  + **Subscribes to the observable with the callback function**
* **A sequence of items that comes asyn over a period of time**

**Promise:**

**-Promise are called only once and will return a single value**

**-not of Lazy**

**-Not cancellable**

**COMPONENT INTERACTION**

* **Share the data between Angular component**
* **Pass data from Parent to Child / Child to Parent**
* **Decorator : @Input, @Output**
* **@Input : Pass data from Parent to Child**
* **@Output: Pass data from Child to Parent**
* **@ViewChild**

**DIRECTIVES:**

* **Component without view/template**

**Type:**

* **Component Directive**
* **Structural Directive: manipulating the DOM elements**
  + **\*ngFor, \*ngIf, ngSwitch…**
* **Attribute Directive : changing the appearance of the DOM elements**
  + **ngStyle, ngClass…**

**LIFE CYCLE OF ANGULAR**

**Constructor**

**OnChange – ngOnChange()**

* **@Input**
* **Whenever there is change in @Input property**

**OnInit – ngOnInit()**

* **When the component is initialized**

**DoCheck – ngDoCheck()**

* **For every change that is detected**

**OnDestroy – ngOnDestroy()**

* **Called before the component is destroyed**

**PIPES:**

* **Transform the data**
  + **Lowercase**
  + **Uppercase**
  + **Date**
  + **Currency**
  + **Json**
  + **Percentage**
  + **Decimal**
  + **Slice**