Angular?

Angular is a framework for building SPA’s

Framework?

Library?

Terminology:

Template – Design

MetaData

Component – Function

Event Binding

Directive

How to Change the Version?

Npm uninstall -g @angular/cli

Npm clean cache –force

Npm install -g @angular/cli

Ng version

Creating a Project:

Ng new project-name

Ng serve / npm start

CSS Grid System / Template Design App.,

1. Css Design
2. Creating Components
3. Routing

Routing:

1. Static Page
2. Dynamic Page

Event Binding:

Event Binding is used to handle the events raised by the user actions. Like., button etc.,

Syntax: <element (event)=function()> </element>

Forms:

1. Template
2. Reactive

MultiFileUpload

Services,

RestApi

RxJs & Observables

Pagination

GetCurrentRoute

ToggleDiv

Directives

1. Component: They are the directive with a Template (view)
2. Structural: Can Change the DOM layout by adding & removing the DOM elements. All the Structural Directives represent \* symbol
3. Attribute:

Examples:

<!-- structural directives -->

<!-- \*ngFor -->

<tr \*ngFor="let customer of customers">

  <td>{{customer.customerNo}}</td>

</tr>

<!-- ngSwitch -->

<div [ngSwitch]="switch\_expr">

  <div \*ngSwitchCase="matchExpr1">cde1</div>

  <div \*ngSwitchCase="matchExpr2">cde1</div>

  <div \*ngSwitchCase="matchExpr3">cde1</div>

  <div \*ngSwitchDefault?></div>

</div>

<!-- ngIf -->

<div \*ngIf="cond">

  statements

</div>

<!-- attribute directives -->

<!-- ngModel -->

<input ngModel #firstName="ngModel" />

<input [(ngModel)] />

<!-- ngClass -->

<div [ngClass]="className"></div>

<!-- ngStyle -->

<div [ngStyle]="{'color': 'blue', 'font-size': '23px'}"></div>

Custom Directive

Pipes

Syntax; Expression | pipeOperator[:pipeArguments]

BuiltIn Pipes

Custom Pipes

Crud Operations

LazyLoading

**Routing**

Ang Router Components:

Router,

Route,

Routes,

RouterOutlet

RouterLink

RouterLinkActive,

ActivatedRoute,

RouterState,

RouterLink Parameters Array

How to Configure Angular Router

Set the <base href>

<base href=’/abou-us’>

Define the routes

Const appRoutes = { path: ‘product’, component: ProductComponent }

Registering the Routes:

Import { RouterModule } from ‘@angular/router’

Imports [RouterModule.forRoot(appRoutes)]

Map Action to Routes:

<a [routerLink]=”[‘/’]”> Home </a>

Access the View

<router-outlet></router-outlet>

**Location Strategies:**

Client Side Routing: <app-root></app-root>

Hash Style Routing

HTML5 Routing

Location Strategy

PathLocationStrategy : http://localhost:4200/product

vs HashLocalStrategy : <http://localhost:4200/#/product>

Hash Style Routing:

Index.,

Id = “about-summary”>

<a name=”routing”> … </a>

/#about-summary

HTML5 Routing:

Var stateObj = { message: ‘…’ };

History.pushState(stateObj, ‘title’, newUrl)

**Route Params;**

{ path: ‘product/:id{/^[0-9]{2,}$/}/:name{/^[a-z]{}$/}/:id3’, component: ProductComponent }

<http://localhost:4200/product>

http://localhost:4200/product/1/5/6

<a [routerLink]=’”[‘/product’, ‘2’, ’4’, ‘’9’]”

http://localhost:4200/635/jkjgw

username: dH0 – 6, 9

/^(?=.\*[0-9) (?=.\*[a-z) (?=.\*[A-Z)[a-zA-Z0-9]{6,9}$/

**Route Guards**

What are Route Guards

Creating Project

Types of Route Guards

How to Create a CanActivate Route Guard

How to Create a CanDeactivate Route Guard

How to Create a Resolver

**Types of Route Guards;**

1. CanActivate: decides if a route (or component) can be activated, like a login system
2. CanDeactivate: decides if a user can navigate away from a route (or component), like asking for confirmation of pending changes
3. Resolve: will delay the activation of a route (or component) until certain tasks are complete, like fetching data from an external source
4. CanActivateChild: is similar to CanActivate but applies to nested routes

Create Guard: ng generate guard custom\_guard\_name

**Local Storage:**

The localStorage object helps us to store data, and the provides 4 methods:

1. setItem
2. getItem
3. removeItem
4. clear

setItem:

localStorage.setItem(‘userName’, ‘John’)

localStorage.setItem(‘user-background’, ‘#f75’)

getItem:

localStorage.getItem(‘userName’)

removeItem:

localStorage.removeItem(‘userName’)

clear:

localStorage.clear()

**Unit Testing:**

A unit test is the process of examining the specific part of the application and make sure it is working correctly

There are 3 ways that we can write tests:

1. unit test
2. integration test
3. end – to – end (e2e) tests

Jasmine: jasmine is an open source behavior – driven testing framework crafter by pivot labs.

It() – declaration of a particular test

Describe() – it’s a suite of all tests

Expect() – expect some value in true from

Karma: it is a test runner tool, creates a browser instance, run tests to provide the expected results.

Unit Testing with Angular Service with HttpClient & HttpTestingController API

Code Snippets:

1. Create the Components for Login, Register, Dashboard, Products, AboutUs, ContactUs
2. Router (Dashboard & Products) – Router Guard
3. Login, Register & ContactUs – 1. Template Driven & Form Validation., 2. Reactive Form & Validation
4. Try to display Google Map with Address
5. LocalStorage – Username, Emailaddress & Password

Create JSON file with group of Users:

[{ username: ‘’, emailAddress: ‘’, password: ‘’ }, { username: ‘’, emailAddress: ‘’, password: ‘’ }, { username: ‘’, emailAddress: ‘’, password: ‘’ }]

Login:

UserName (or) EmailAddress

Password

Demo1, Demo1

Dashboard., - Profile Photo & Preview

Products:

RestApi.,

CRUD

Product Title

Product Description

Product Images (\*)

Tabular.,

All the Product

Register:

First Name (a, A, )

Last Name (a, A, )

User Name (a, N, 2 – 5, 8)

Email Address (@, .)

Password (a, A, 3, $ - 8, 12) & Password (below that password input element, create a Button., when user clicks on that button (Generate Password)

DOB (Calender)

Gender

Country – RestApi.,

Create a JSON File with group of Countries List., and access that JSON file into the Country Dropdown