**ANGULAR**

**Single Page Application**

A single-page application (SPA) is a website design approach where each new page's content is served not from loading new HTML pages but generated dynamically through JavaScript's ability to manipulate the DOM elements on the existing page itself.

Angular is SPA Framework or platform which is built using HTML and TypeScript.

Architectural Features:-

1. Components

Combination of DATA and LOGIC.

Consists of HTML Template to render DATA using LOGIC onto the web browser.

Every application has a ROOT component followed by different components to render different parts of application onto the browser.

1. HTML Template

HTML Template uses two other cool features

1. Angular Directives – to define logic
2. Data Binding – to bind data with HTML DOM Element
3. Event Binding – button clicks, or when user places mouse over text, etc.
4. Property Binding – bind the HTML input properties to the model in our angular application component.
5. Service

To fetch data from backend.

We can use Dependency Injection in Components to inject a service in the component and component can use the service to fetch the data and render it to the UI.

1. Routing

Navigating from one point to another

1. Versions

2, 4, 5, 6, 7, 8, 9

Angular 2 is a major upgrade over Angular 1.

Later versions are just some minor upgrade over the previous version.

**Installing Angular**

1. Install node
2. In the command line, type npm install –g @angular/cli
3. Install Yarn(this is just like npm(node packaging manager))
4. Install VSCode
5. Install MYSQL Workbench

**Creating a project**

ng new my-app

cd my-app

ng serve -o

**ng new**

The Angular CLI makes it easy to create an application that already works, right out of the box. It already follows our best practices!

**ng generate**

Generate components, routes, services and pipes with a simple command. The CLI will also create simple test shells for all of these.

**ng serve**

Easily test your app locally while developing.

**Components**

app.module.ts is the most important file where all the imports(external and internal) are defined.

import { BrowserModule } from '@angular/platform-browser';

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

import { AppComponent } from './app.component';

import { ProductComponent } from './product/product.component';

@NgModule({

  declarations: [

    AppComponent,

    ProductComponent

  ],

  imports: [

    BrowserModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

app.component.ts is the one where we define out component and class associated with it.

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'angularBasics';

}

app.component.html and app.component.css are used to define html and css respectively for the component.

**ngFor Directives**

products: Object[];

  constructor() {

    this.products = [

      {

        "id":"1",

        "name":"Mackbook Pro"

      },

      {

        "id": "2",

        "name":"ROG"

      }

    ];

  }

  public getProducts() {

    return this.products;

  }

<div \*ngFor="let product of getProducts">

    <h1>{{product.id}}</h1>

    <h1>{{product.name}}</h1>

</div>

**Input Property Binding & Ouput Event Binding**

<b [hidden]="hide">{{getSellers()}}</b>

<input type="button" value="Toggle" (click)="hide=!hide">

**Directives**

Directives are attributes that we add to our html elements which will dynamically affect the HTML DOM that gets generated when our page is rendered.

1. **ngIf**

It is used to display HTML element conditionally. That if the result on right is true, then the element is displayed otherwise removed

<b \*ngIf=”true”>My Text</b>

Here, it is hardcoded to true, but we can also use a javascript variable or function.

1. **ngSwitch**

Similar to switch in JavaScript

<div [ngSwitch]=”mycase”>

<div \*ngSwitchCase=”one”>

<div \*ngSwitchCase=”two”>

<div \*ngSwitchDefault>

1. **ngFor**

Similar to for in JavaScript

<div \*ngFor=”let c of courses”>

<b>c</b>

</div>

1. **ngNonBindable**

This will not compile the part where it is used.

That part will be displayed as it is.

1. **ngStyle**

[style.<cssproperty>]=”value”

[ngStyle]={‘key:’value’}

<table>

    <thead>

        <tr>

            <td [ngStyle]="{'color':'red', 'background-color':'blue'}">First Name</td>

            <td>Last Name</td>

            <td>Age</td>

        </tr>

    </thead>

    <tr \*ngFor="let s of students">

        <td [style.background-color]="'red'">{{s.fname}}</td>

        <td [style.color]="'green'">{{s.lname}}</td>

        <td [style.color]="'blue'">{{s.age}}</td>

    </tr>

</table>

1. **ngClass**

[ngClass]=”{tdata:true}”

tdata is CSS property block.

useTData: boolean;

  classObj: object;

this.useTData = true;

this.classObj = {

  tdata: this.useTData

};

.tdata{

    font-size:80px

}

<td [style.color]="color" [ngClass]="classObj">{{s.age}}</td>

**Services**

**Dependency Injection**

Dependency injection is a technique whereby one object supplies the dependencies of another object. A "dependency" is an object that can be used, for example as a service.

To do dependency injection in angular, first register any dependency in NgModule.

Ex:-

Import {LoginService} from ‘../services/login.service’;

@NgModule({

Providers:[

LoginService

]

})

Second, inject the dependency

Ex:-

Import {LoginService} from ‘../services/user.service’;

Class LoginComponent{

Constructor(private loginService:LoginService){

}

}