Angular 2 :

**Run by : localhost:4200**

For single page application :

Angular2 cli .With cli it is easy. Need node js. Npm manager for various package manager.

1)Install angular-cli through the npm

npm install angular-cli –g

2)Create new project with name first-app inside the folder

ng new first-app

3)Start the development server :

ng serve

It also build our application.Angular 2 uses typescript.It converts the typescript back to the javascript. **Note** : Browser can not be able to run the typescript

Structure of the folder.

1. src/app folder contains all the ts files (Components).Type script
2. app.component.ts

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

@Component({

selector: 'app-root',

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

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

})

export class AppComponent {

title = 'app works!';

}

AppComponent is the typescript class.The @component decorator is attached to this class. The @Component decorator is imported from the angular2/core. It allows us to transform the normal typescript class into something else.Each component in our application need some html to display.

**Note** :stylesUrl is optional not the template.It is array so, we can use multiple files here for the css.

**Selector** : Allows us to use the component inside the html .Our custom html element. It decides how we want to insert the component to our html code.The**app-root** here refers to the html element app-root.It tells the angular 2 what to do when it enconter’sthis  **app-root** inside the html **.**

**Different ways the selector are used :**

a)**#app-root**  - This is used whenever we are using the app-root as an id.

b)**.app-root** – This is used when we are using the app-root as an class of html element.ex : <span class=”app-root”></span>

Above we can see the decorator with @ sign.

Note : Angular 2 use the components to render our web application.each component not only contains the typescript but also have some html to display.

Template of AppComponent is stored in **app.component.html**

Cssof AppComponent is stored in app.component.css.It is array type means it can contain more then one css.

**Now let’s see the html fir this file.**

<h1>

{{title}}

</h1>

{{}} is known as the string interpolation.It can display any property of our component dynamically.Here title is the property of the AppComponent class.

Inside the app folder we do have the spec file that is usually for the unit testing.

The different files are :

a)app.component.html – This is template for our component

b)app.component.css – This is css for out component

c)app.component..ts – This is out component file with ts extension that is typescript.

d)index.ts – This file keeps trak of all the files present inside the app fiolder. So, that the export from our folder can be easier.

**Environment Folder**

This folder contains some of the configurations which is needed for the compilation of our app. The default here is fine so, we not need to do any change here.Same for the other files. These are the files used by our cli(**Angular-cli**) to build our application.

The main files are :

a)**main.ts** : This is file that just starts our angular 2 application.

2)**styles.css** - This file is used to apply the global css. That is applied to our whole application. While the styles applied to our app folder is applied directly to that component only.

3)**angular-cli.json** - This files allow us to just change the setting of our angular-cli. For example where we want to store our source file etc.

4)**node\_modules** - contains all the dependencies of our project, We never change it . It is simply managed by our cli.

5)**dist folder**  - This is the folder where we never do any of the change. All the compiled codes are stored here by the cli. This we can deploy to the server for example.

**Note :**We mainly works inside the app folder.

**Typescript**

With creating the application with the help of angular-cli , the typescript get’s automatically installed but we can install it globally with the help of npm tool.

**npm install typescript –g**

**index.html –** Angular2 is a single page application.So it is not the component that is loaded but it is the index.html that is loaded.It the file that is loaded by our server.It is the single page application.

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>FirstApp</title>

<base href="/">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" type="image/x-icon" href="favicon.ico">

</head>

<body>

<app-root>Loading...</app-root>

</body>

</html>

Above we can see **<app-root>Loading...</app-root>**.This is our component

**How the angular 2 app gets started ?**

Ans : There is no script we are seeing here.

Let us see our browser and see the source code there. We can see the 3 scripts download here.

a)inline.js

b)styles.bundle.js

c)main.bundle.js - **Imp**

These files are added during the compile time.**Main.bundle.js** contains all our code + angular2 code + all 3rd party package code. This also starts our angular application.

**Now, how it is started (main.ts)**

We can see the main.ts :It is the first file that get started by the bundle.It bootstrap our application.It starts our application by below line.

platformBrowserDynamic().bootstrapModule(**AppModule**);

**How the angular2 come to know which component to load ?**

Angular does not load the whole component. It just see it inside the AppModule.

Where the declaration of component is there.Just see the app.module.ts.

],

providers: [],

**bootstrap: [AppComponent]**

It bootstrap the AppComponenthere.This tells the angular 2 that this component is the root component of the application. This is also present inside the index.html

**Understanding the AppModule Component**

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

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

import { FormsModule } from '@angular/forms';

import { HttpModule } from '@angular/http';

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

@NgModule({

declarations: [**//Tells which directive we use in our application.Directives are the instructions that tells angular2 what to do.**

AppComponent

],

imports: [ /**/Tells which other modules do I use in our application.**

BrowserModule,**//These are predefined modules.**

FormsModule,**//when working with input fields**

HttpModule**//When working with http access**

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

Here we have **@NgModule**decorator that we are importing from the @angular/core. It tells what belongs to our app.

**Using template and styles :**

The templateUrl is replaced by the **template**

Can use template instead of templateUrl.

**template** : `

Multiple lines can come here.

Line 1

Line 2

`,

**styles** :[`

h1 {

color : red

}

`]

Q1)Create an task for assigning the styles for h1(red),h2(green),h3(blue).Apply the attribute values like first,second,third

**Creating the newcomponent :**

**Through the cli command.**

ng generate component other

If we want this component to show on the screen.Do small change in the **app.module.ts**: bootstrap: [**OtherComponent**]**.** Also we need to change the selector inside the index.html. **<app-other>Loading...</app-other>**

**App component is already running.**So we will use the other component inside the AppComponent.

template: `

<h1>{{first}} item</h1>

<h2>{{second}} item</h2>

<h3>{{three}}</h3>

<app-other></app-other>

`

Multiple lines template :

Q1)Create 4 modules with name refrigerator , television , radio , microvan

Use then inside the appcomponent.

Q2)Again same as above , but use 2-2 modules with id reference and class reference.

**Note : Give the selectors unique name…**fa-refrigerator

**Custom ng generate command**

ng g c another –flat –inline-styles –inline-template

or

ng g c another –flat –is –it

q1)generate modules like apple , banana , strawberry without the template and the css

**View encapsulation and binding**

The problem is that when we applied the css for the h1 tag inside the main component . It does not gets applied to other component h1 element.

This problem is solved by the **view encapsulation**

**The shadow DOM** : It means that the html element has it’s different DOM behind the scene. So, all the **h1** element has there own shadow DOM there.Inspect the page and show it for the **h1** tag.

\_ngcontent-unc-0

**Note :**We can see the h1 declaration in the head tag of the inspect page.

**Inserting content with ng-content**

So, inside the radio template define <section><ng-content></ng-content></section>.Also define the css for that like border : 1px solid black

Now, whatever we pass through the main component html gets displayed here.

Like :

template: `

<h1>{{first}} item</h1>

<h2>{{second}} item</h2>

<h3>{{three}}</h3>

<app-other></app-other>

<app-refrigerator></app-refrigerator>

<app-television></app-television>

<app-radio>

radioradio

</app-radio>

<app-microvan></app-microvan>

`

Radio radio gets displayed inside the article tag.

**Q1)create an application .Make sub modules as Sachin , dravid , ganguly. Assign the ng-content to each component template .Also assign the div css.At last assign the values from the main component html file.**

**Q2)Create an application using the bootstrap containing the 3 modules .Login , Registration , Forgot Password.**

Hint :

<div class=”container”>

<div class=”row”>

<div class=”col-lg-6”>

<app-login></app-login>

</div>

<div class=”col-lg-6”>

<app-register></app- register >

</div>

<div class=”col-lg-12”>

<app-forgot></app-forgot>

</div>

</div>

</div>

Thus we can make reusable containers.If we use in this way

<app-radio>

radioradio

</app-radio>

It will be red taking the css for the appComponent.

**Data binding :**

String interpolation {{}}

Property binding :<button [disabled] = “expression resolving to required value type”></button>. So disabled is the property of the button.

Event binding : <button (click)=”expression handling the event:”>

Click is the event for the button.

Two way data binding :

<input [(ngModel)]=”bound model (eg object)”>

**Note :**Angular 2 does not supports the 2 way data binding .It is uni directional. But by using [()] Property binding + event binding we can have the 2 way data binding.

**Property binding and event binding :**

Dom(**Document object model**)property : It is the native property already exist in the DOM. Like src , click event. There are not of angular 2

<img [src]=””> : Property

<button (click)=””> : Event

Q1)Create a component append click event to the button

Directive Property : These are of angular 2.

<div [ngClass]=””> : This directive is used to add the css class for the something element. : Property

<button (ngSubmit)=””> : event

Q1)Write a script for the directive binding for the **ngClass , ngStyle**.Also show on the inspect element.

Component Properties : Property set up in the class of our component. Class can be bound or can be events that can be listen from the outside.

<cmp [initObj] > : Property

<cmp (eventDom)> : Event

Q1)Create a new component and pass the value from the outside.Take attributes as **first\_name,last\_name,email, contact , age ,city , address.** Display these items in the template .It is property binding from the outside.

Hint :**@Input() first\_name : String**

<my-app [contact]=’9730232815’></my-app>

Q2)Create a component to create own event and listen to that.

Hint :@Output() clicked = new EventEmitter<string>()

**Some important hacks about the custom event binding**

a)@Output() clicked = new EventEmitter<string>()

Or

@Output(‘tigor’) clicked = new EventEmitter<string>() : Foralternate name of clicked

this.clicked.emit('Me Mohan here');

b)Instead of using the decorators @ we can replace in with the inputs[], outputs[]

**Note :**

All the above are targetable through the property and event binding . Ladder to directive and component properties we may use some already build in one or some of our custom binding.

For Property binding:@Input()propertyName : String

For event binding:@OutputeventName = new EventEmitter()

**Two way data binding :**

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

@Component({

selector: 'app-two-way-binding',

template: `

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

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

`,

styles: []

})

export class TwoWayBindingComponent {

person = {name : 'Mohan',age : 27}

}

**Directives in Angular2**

Component are also known as the directive .They are simply the instructions that tells angular 2 what to do at certain place.It tells render my view at this place.Also helps to change the normal appearance of the **dom**.

**Example1** :

<ul>

<li \*ngFor=”let item for items”>{{item}}</li>

</ul>

Instruction :

Loop through all the items. In each iteration assign the current element to variable called item. Also replicate <li> element for each iteration.

**Example2** :

<div>

<my-component></my-component>

</div>

Instruction : Create a component of selected Type and render the view of the component into this place.

Besides the components we have the Attribute and the structural directive.

**Attribute directive** : They interact with the element to which they are applied to. They act as an attribute.

Ex : [ngClass],[ngStyle]

[] is only for the property binding.

**Q1)Apply ngClass directive . Set border and background class**

[ngClass]=”{border:true,background:true}”

Or

<div [class.border]=”true”></div>

Hint : .border { border : 3px solid blue; }

.background { background-color : red}

Div {width : 100px ; height : 100px}

**Q2)Apply ngStyle directive .**

<div [ngStyle]=”’background-color’:red”></div>

**Structural directive** :

They are named because they change the structure of the DOM element/Html code.

Ex: \*ngFor,\*ngIf

**Note** : Create a new component : ng new directives –directory pankaj\_directive –prefix dir

**Custom attribute directive**

Q1)Create with angular-cli : ng g d **highlight**

**<div appHighlight>All is well</div>**

import { Directive,ElementRef } from '@angular/core';

@Directive({

selector: '[appHighlight]'

})

export class HighlightDirective {

constructor(private ele : ElementRef)

{

this.ele.nativeElement.style.backgroundColor = 'green';

}

}

**Q2)Create a custom directive with mouseenter and mouseleave effect.**

import { Directive,HostBinding,HostListener } from '@angular/core';

@Directive({

selector: '[appHover]'

})

export class HoverDirective {

@HostListener('mouseenter') mouseenter(){

this.backgroundColor = 'green';

}

@HostListener('mouseleave') mouseleave(){

this.backgroundColor = 'red'

}

private backgroundColor = 'green';

@HostBinding('style.backgroundColor') get setColor(){

return this.backgroundColor;

}

constructor() { }

}

**<div appHover>Albert Einstein</div>**

**Directive passing value from the outside**

<div appHover [backgroundColor]="'blue'">Albert Einstein</div>

@HostListener('mouseenter') mouseenter(){

this.backgroundColor = 'green';

}

@HostListener('mouseleave') mouseleave(){

this.backgroundColor = 'red'

}

@Input() backgroundColor = 'green';

@HostBinding('style.backgroundColor') get setColor(){

return this.backgroundColor;

}

**Q3)Create a custom directive with mouseenter and mouseleave effect on paragraph text color.**

Again pass the mouseoverColor,mouseoutColor from external.

**Structural directive :**

\*ngFor , \*ngIf , \*ngSwitch

Q1)Create a button and hide/show some text.

Q2)Loop through items array using \*ngFor.Ul tag

Hint : let item of tems

<li \*ngFor="let item of items;let i = index;">

{{i}} - {{item}}

</li>

Q3)Example for \*ngSwitch

<div [ngSwitch]="value">

<p \*ngSwitchCase="100"> 100 is here </p>

<p \*ngSwitchCase="200"> 200 is here </p>

<p \*ngSwitchDefault>Other</p>

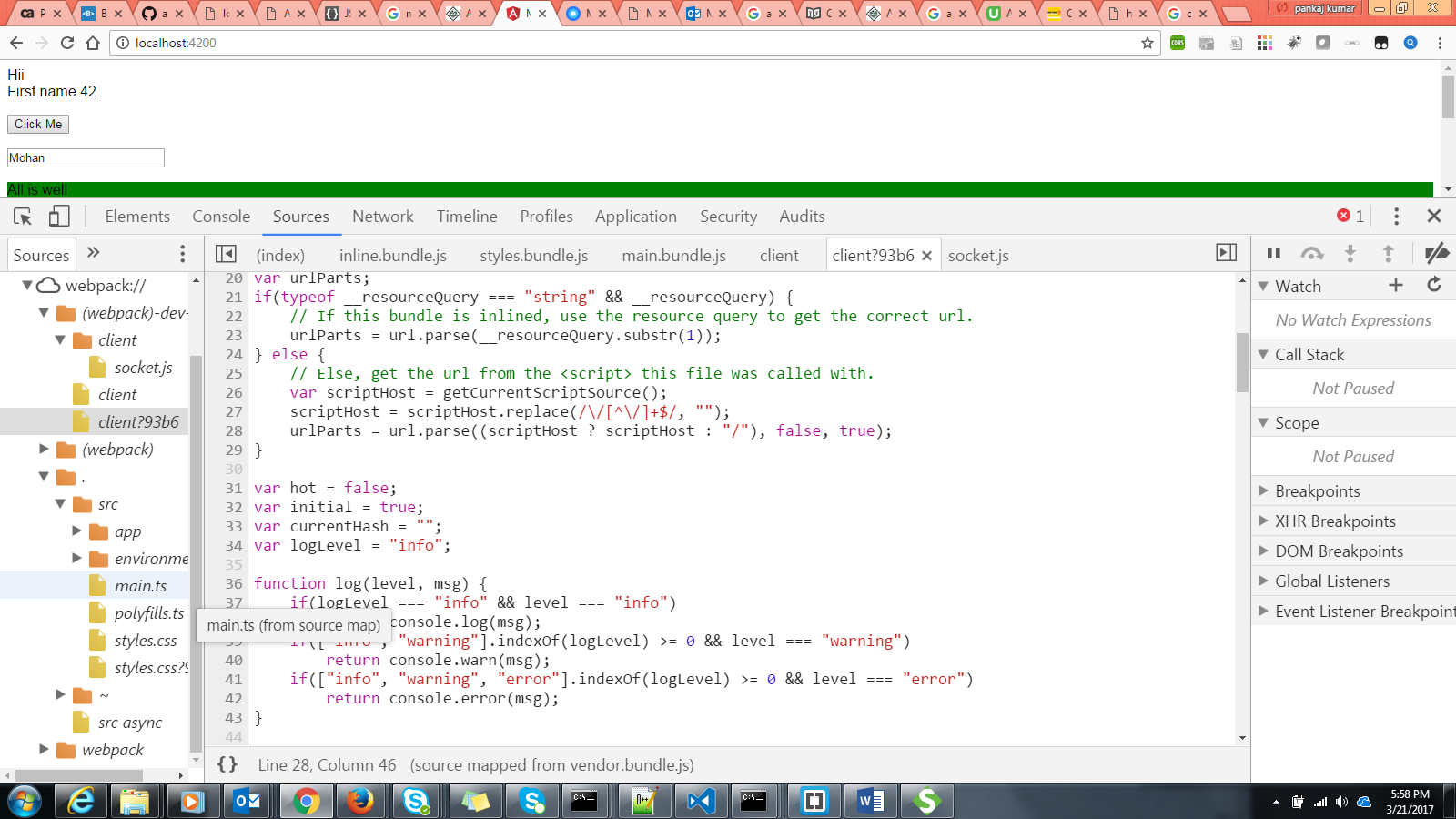
</div>

Creating custom structural directive :

**Note : Structural directive contains the \***

**Debugging the angular2 application**

Inspect the chrome . Go to sources tab.



Another toll we can from <https://augury.angular.io/>. Install in chrome

How to use : Inspect the chrome and go to augury.

Q1)**Take a for loop to initialize an string array.**

marks : string[] = [];

for(let i=0;i<=10;i++)

{

this.marks.push('The record is '+i)

}