**Software Installation and Project Set Up:**

Using Command Line:

* Downloas Node Js
  + https://nodejs.org/en/download/
* **Step-1:** set the path in envirnment variables
  + C:\Users\t-Ravindra1\AppData\Roaming\npm
* **Step-2:** chech the version of software
  + node -v
* **step-3:**check the npm version
  + npm -v
* **Step-4:** To install angular 6
  + npm install -g @angular/cli
* **Step-5:** Create project (Project name: Angular6App (any name))
  + ng new Angular6App
* **Step-6:** Download Visual Studio
  + https://code.visualstudio.com/
* **Step-7:**Open project in Visual Studio (from your location: c:C:\Users\t-Ravindra1)
* **Step-8:** Compile our project (C:\Users\t-Ravindra1\Angular6App>)
  + ng serve
* **Step-9:**See the output (Run on web browser)
  + http://localhost:4200/

**Introduction:**

* The first version that was released is Angular 1, which is also called AngularJS.
* Angular 6 released in May 2018 .
* The structure of Angular is based on the components/services architecture.
* The structure of AngularJS was based on the model view controller.

**Features of Angular 6:**

1. **ng-add:** to quickly add application features to make application a progressive web apps.
2. **ng-update:** to migrate from previous version to current version.
3. Supports creating **custom UI elements** without need of angular material library.
4. Supports responsive **web design layouts**.
5. Supports overlay packages to **create pop-ups**.
6. Allows Angular Components to be published as Web Components which can then be used in any **HTML page**.
7. Added: **mat-tree**, a styled version and **cdk-tree**, a unstyled version

**Components:**

1. app.component.css
2. app.component.html
3. app.component.spec.ts
4. app.component.ts
5. app.module.ts

|  |  |  |
| --- | --- | --- |
| **S.No** | **Component Name** | **Use** |
| 1 | app.component.css | css file for the new component is created. |
| 2 | app.component.html | html file is created. |
| 3 | app.component.spec.ts | this can be used for unit testing. |
| 4 | app.component.ts | we can define the module, properties, etc. |
| 5 | app.module.ts | It has some libraries which are imported and also a declarative which is assigned the appcomponent. |

|  |  |  |
| --- | --- | --- |
| **S.No** | **app.component.ts** | **app.module.ts** |
| 1 | import { Component } from '@angular/core'; | import { BrowserModule } from '@angular/platform-browser';  import { NgModule } from '@angular/core';  import { AppComponent } from './app.component'; |
| 2 | @Component({  selector: 'app-root',  templateUrl: './app.component.html',  styleUrls: ['./app.component.css']  }) | @NgModule({  declarations: [  AppComponent  ],  imports: [  BrowserModule  ],  providers: [],  bootstrap: [AppComponent]  }) |
| 3 | export class AppComponent {  title = 'Angular6Project';  } | export class AppModule { } |

**@NgModule:**

@NgModule and contains an object which has declarations, import s, providers and bootstrap.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Object name** | **Description** | **Example** |
| 1 | Declaration | It is an array of components created. If any new component gets created, it will be imported first and the reference will be included in declarations. | declarations: [  AppComponent,  NewCmpComponent  ] |
| 2 | Import | It is an array of modules required to be used in the application. It can also be used by the components in the Declaration array. For example, right now in the @NgModule we see the Browser Module imported. In case your application needs forms, you can include the module. | import { FormsModule } from '@angular/forms';  The import in the @NgModule will be like the following −  imports: [  BrowserModule,  FormsModule  ] |
| 3 | Providers | This will include the services created. | - |
| 4 | Bootstrap | This includes the main app component for starting the execution. |  |

**Note-1:**

**<app-root></app-root>**

* This is the root tag created by the Angular by default. This tag has the reference in the main.ts file.

**Note-2:** To create your own component

**Command:** ng generate component employee

**Example:**

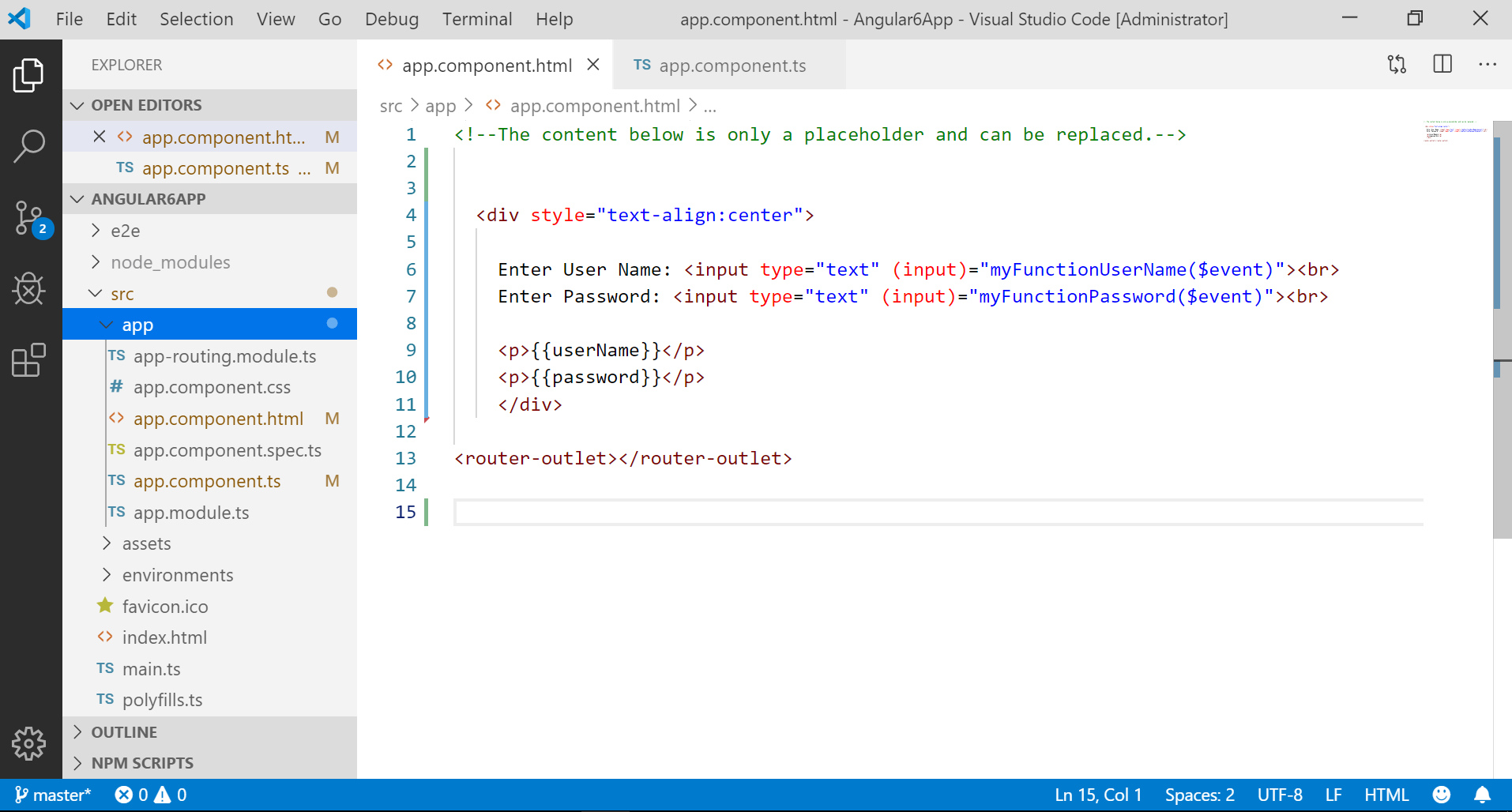
employee.component.html

employee.component.spec.ts

employee.component.ts

employee.component.css

**Derectory Structure in Visual Studio:**



**Topic: Data Binding:**

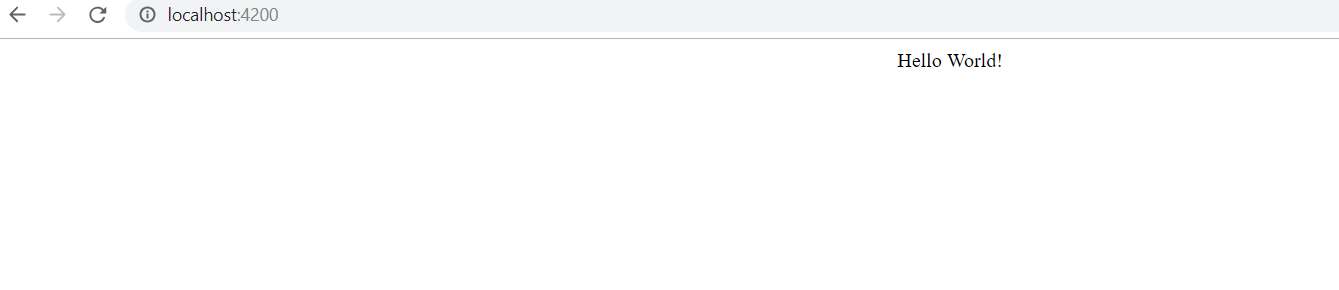
**Example-1:**

**app.component.ts**

|  |
| --- |
| import { Component } from '@angular/core';  @Component({  selector: 'app-root',  templateUrl: './app.component.html',  styleUrls: ['./app.component.css']  })  export class AppComponent {  title = 'Hello World';  } |

**app.component.html**

|  |
| --- |
| <!--The content below is only a placeholder and can be replaced.-->  <div style="text-align:center">    {{ title }}!    </div>    <router-outlet></router-outlet> |



**Example-2: Display Employee Details uisng Data Binding**

**app.component.ts**

|  |
| --- |
| export class AppComponent {  employeeId = 1020;  employeeName='Ravindra';  salary=1800000.00;  } |

**app.component.html**

|  |
| --- |
| <div style="text-align:center">    Employee Id: {{ employeeId }}<br>  Employee Name: {{ employeeName }}<br>  Employee Salary: {{ salary }}<br>  </div> |

**Example-3: Data Binding using Constructor**

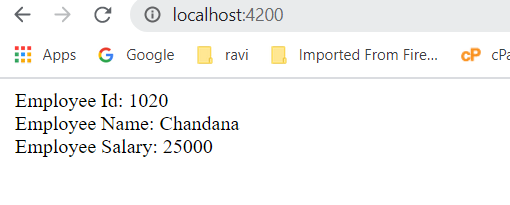
**app.component.ts**

|  |
| --- |
| export class AppComponent {        employeeId:Number;      employeeName:String;      salary:Number;      constructor(){        this.employeeId=1020;        this.employeeName="Chandana";        this.salary=25000.00;      }  } |

**app.component.html**

|  |
| --- |
| <body>    Employee Id: {{employeeId}}<br>    Employee Name: {{employeeName}}<br>    Employee Salary: {{salary}}<br>  </body> |

**Output:**



**Data Binding using Class and Object:**

**Example-4: Data Binding using class and Object**

**Step-1:**

|  |
| --- |
| **C:\Users\ravichinni86\Angular6App>ng generate class Employee** |

**Step-2: C:\Users\ravichinni86\Angular6App\src\app\employee.ts**

|  |
| --- |
| export class Employee {  employeeId:number;  employeeName:string;  salary:number;  constructor(employeeId:number,employeeName:string,salary:number){  this.employeeId=employeeId;  this.employeeName=employeeName;  this.salary=salary;  }  } |

**step-3: C:\Users\ravichinni86\Angular6App\src\app\app.component.ts**

|  |
| --- |
| import { Component } from '@angular/core';  **import { Employee } from './employee';**  @Component({  selector: 'app-root',  templateUrl: './app.component.html',  styleUrls: ['./app.component.css']  })  export class AppComponent {    empObject=new Employee(1020,"James",25000.00);    } |

**Step-4: C:\Users\ravichinni86\Angular6App\src\app\app.component.html**

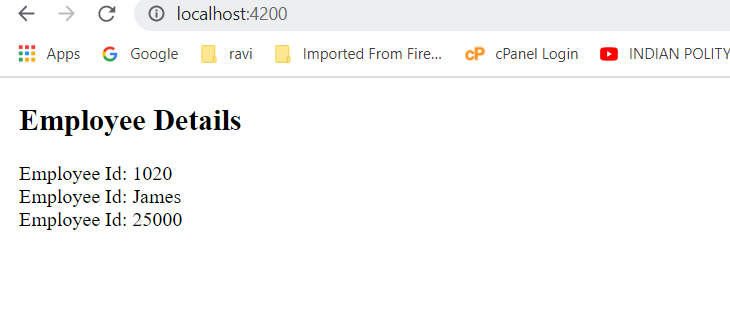
|  |
| --- |
| <html>  <head></head>  <body>  <h2>Employee Details</h2>  Employee Id: {{empObject.employeeId}} <br>  Employee Id: {{empObject.employeeName}} <br>  Employee Id: {{empObject.salary}} <br>  </body>  </html> |

**Step-5: Compile**

|  |
| --- |
| **C:\Users\ravichinni86\Angular6App>ng serve** |

**Step-6: Run**

|  |
| --- |
| **http://localhost:4200/** |



**Example-5: Drop Down box using Data Binding**

**app.component.ts**

|  |
| --- |
| export class AppComponent {  title = 'Angular6App';  department = ["java", ".net", "testing"];    } |

**app.component.html**

|  |
| --- |
| <div> Department :  <select>  <option \*ngFor = "let dept of department ">{{dept}}</option>  </select>    </div> |

**Topic: Event Binding**

When a user interacts with an application in the form of a keyboard movement, a mouse click, or a mouseover, it generates an event. These events need to be handled to perform some kind of action. This is where event binding comes into picture.

**Example-1:**

**app.component.ts**

|  |
| --- |
| export class AppComponent {  country = ["India", "USA", "UK"];  myFunction(event: any) {    alert("Data has been successfully submited");  console.log(event);  }  } |

**app.component.html**

|  |
| --- |
| <div>  <select>  <option \*ngFor = "let i of country">{{i}}</option>  </select>  </div>  <div>  <button (click)="myFunction($event)">submit</button>  </div> |

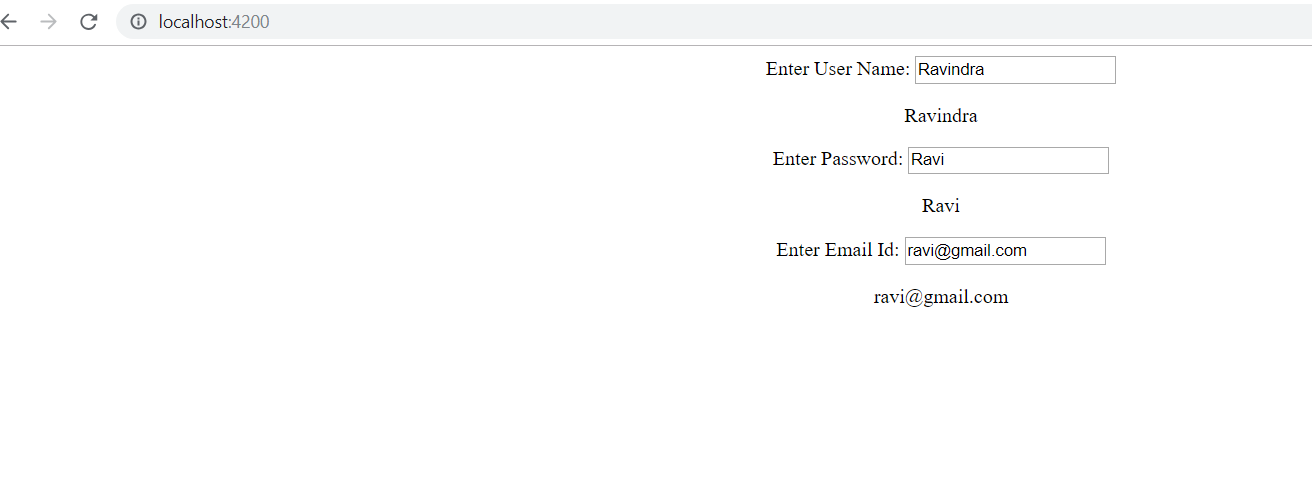
**Example-2: username,password etc..**

**app.component.ts**

|  |
| --- |
| export class AppComponent {    userName = '';  password = '';  emailId='';    myFunctionUserName(event: Event) {    this.userName = (<HTMLInputElement>event.target).value;  }  myFunctionPassword(event: Event) {  this.password = (<HTMLInputElement>event.target).value;  }  myFunctionEmailId(event: Event) {  this.emailId = (<HTMLInputElement>event.target).value;  }  } |

**app.component.html**

|  |
| --- |
| <div style="text-align:center">  Enter User Name: <input type="text" (input)="myFunctionUserName($event)">  <p>{{userName}}</p>  Enter Password: <input type="text" (input)="myFunctionPassword($event)">  <p>{{password}}</p>  Enter Email Id: <input type="text" (input)="myFunctionEmailId($event)">  <p>{{emailId}}</p>  </div> |



**Topic: Templates**

Angular 6 uses the <ng-template> as the tag similar to Angular 4 instead of <template> which is used in Angular2. The reason Angular 4 changed <template> to <ng-template> is because there is a name conflict between the <template> tag and the html <template> standard tag. It will deprecate completely going ahead.

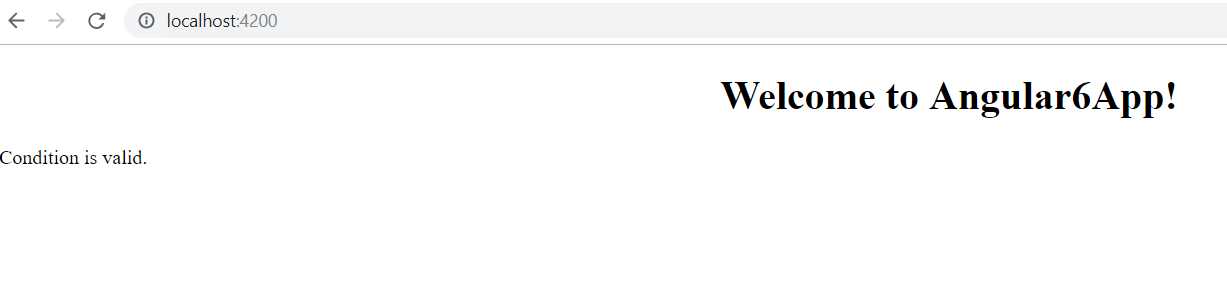
**Condition Validation (If)**

**app.component.ts**

|  |
| --- |
| export class AppComponent {  title = 'Angular6App';  country = ["India", "USA", "UK"];  isavailable = true;  } |

**app.component.html**

|  |
| --- |
| <div>  <span \*ngIf = "isavailable">Condition is valid.</span>  <!--over here based on if condition the text condition is valid is displayed.  If the value of isavailable is set to false it will not display the text.-->  </div> |



**If..else**

**Case-1: Invalid**

**app.component.ts**

|  |
| --- |
| export class AppComponent {  title = 'Angular6App';  country = ["India", "USA", "UK"];  isavailable = false; //invalid  } |

**app.component.html**

|  |
| --- |
| <div>  <span \*ngIf = "isavailable; else condition1">Condition is valid.</span>  <ng-template #condition1>Condition is invalid</ng-template>  </div> |



**Case-2: Valid (if..then..else)**

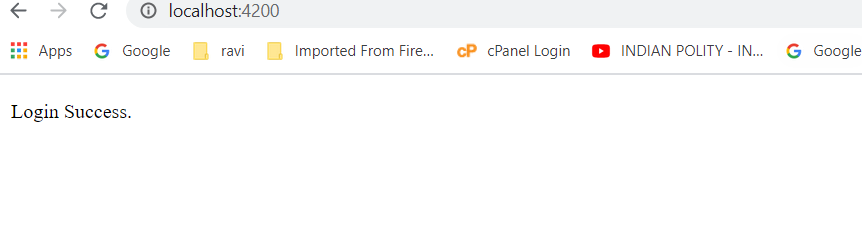
**app.component.ts**

|  |
| --- |
| export class AppComponent {      isLoggedIn=false;    } |

**app.component.html**

|  |
| --- |
| <ng-container \*ngIf="isLoggedIn; then loggedIn; else loggedOut"></ng-container>        <ng-template #loggedIn>Login Success.</ng-template>      <ng-template #loggedOut>You have Logout..Please Login</ng-template> |

**Output:**



**Example-5: Eligible for Vote or not?**

**app.component.ts**

|  |
| --- |
| export class AppComponent {     age=15;    } |

**app.component.html**

|  |
| --- |
| <ng-container  \*ngIf="age < 18; else elseBlock">Not Eligible to vote.</ng-container>       <ng-template #elseBlock>Eligible to vote.</ng-template> |

**Output:**

**Constructor:**

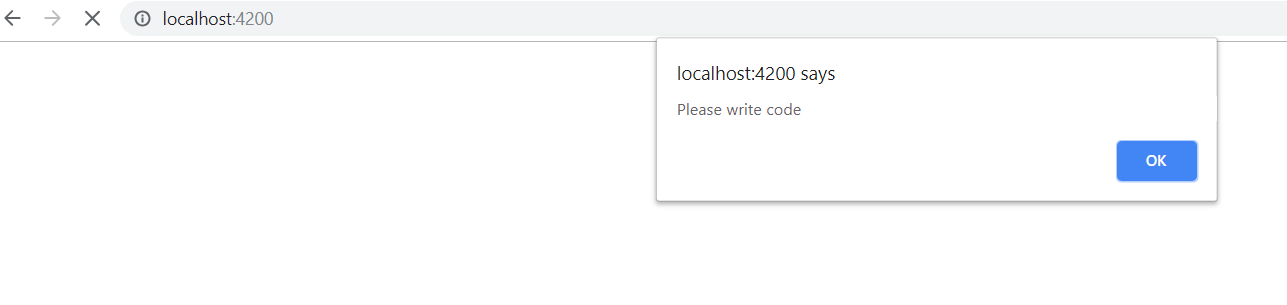
**app.component.ts**

|  |
| --- |
| export class AppComponent {    constructor(){  alert("Please write code");  }    } |

**app.component.html**

|  |
| --- |
| <div style="text-align:center">      </div> |

After refresh,you will get the following output.



**Directives:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Directive Name** | **Use** |
| 1 | Component Directives | These form the main class having details of how the component should be processed, instantiated and used at runtime. |
| 2 | Structural Directives | A structure directive basically deals with manipulating the dom elements. Structural directives have a \* sign before the directive. For example, **\*ngIf and \*ngFor**. |
| 3 | Attribute Directives | Attribute directives deal with changing the look and behavior of the dom element. You can create your own directives(Custom Directives). |

**Pipes:**

Pipes were earlier called filters in Angular1 and called pipes in Angular 2 onwards.

The | character is used to transform data. Following is the syntax for the same

**Example:**

{{ userName | lowercase}}

It takes integers, strings, arrays, and date as input separated with | to be converted in the format as required and display the same in the browser.

**Example-1:**

**app.component.ts**

|  |
| --- |
| export class AppComponent {  message = 'Hello World';  } |

**app.component.html**

|  |
| --- |
| <b>{{message | uppercase}}</b><br/>  <b>{{message | lowercase}}</b> |

Angular 6 provides some built-in pipes. The pipes are listed below:

1. Lowercase
2. Uppercase
3. Date
4. Currency
5. Json
6. Percent
7. Decimal
8. Slice

Pipes: (Filters)

1. Lowercase

2. Uppercase

3. Date (date:'d/M/y')

4. Currency (currency:"USD")

5. Json

6. Percent

7. Decimal (number: '3.4' )

8. Slice (slice:2:4) (Ex: state=["AP","TS","MH","KA","DL"];)

Ex: {{ propetyname |pipe}}

**Example-1:**

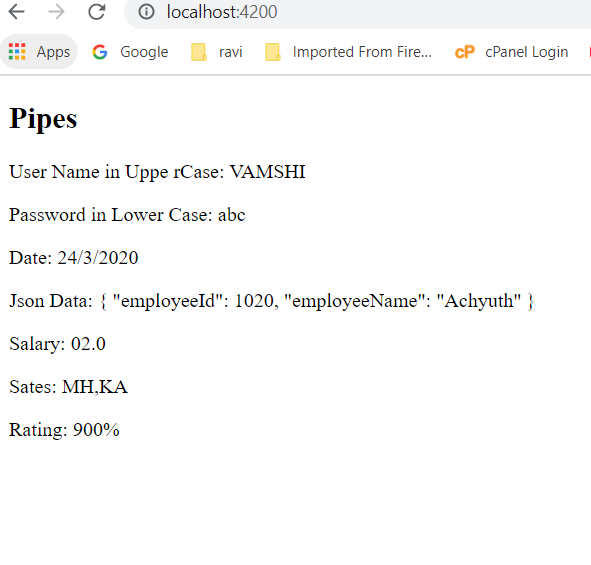
**app.component.ts**

|  |
| --- |
| export class AppComponent {     username="Vamshi";   password="ABC";   date=new Date();   jsonData={employeeId:1020,employeeName:"Achyuth"};   state=["AP","TS","MH","KA","DL"];   salary=2;   rating=9;  } |

**app.component.html**

|  |
| --- |
| <body>      <h2>Pipes</h2>      <p>User Name in Uppe rCase: {{username|uppercase}}</p>      <p>Password in Lower Case: {{password|lowercase}}</p>      <p>Date: {{date|date:'d/M/y'}}</p>      <p>Json Data: {{jsonData|json}}</p>      <p>Salary: {{salary|number: '2.1'}}</p>      <p>Sates: {{state|slice:2:4}}</p>      <p>Rating: {{rating|percent}}</p>  </body> |

**OUTPUT:**



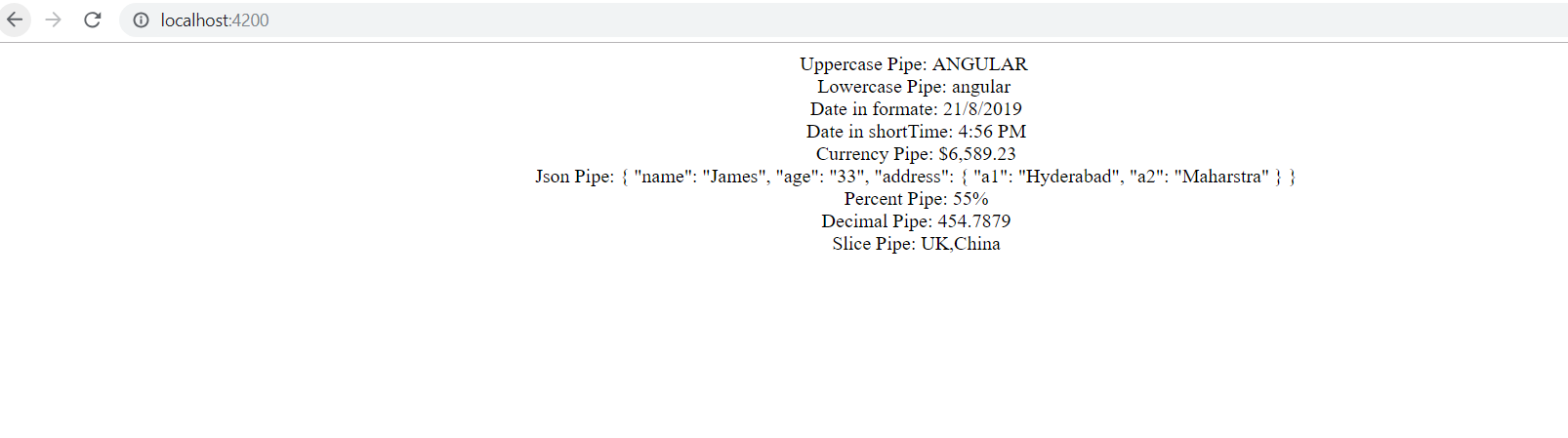
**Example-2:**

**app.component.ts**

|  |
| --- |
| export class AppComponent {  title = 'Angular';  todaydate = new Date();  jsonval = {name:'James', age:'33', address:{a1:'Hyderabad', a2:'Maharstra'}};  country=["India","USA","UK","China","Japan"];    } |

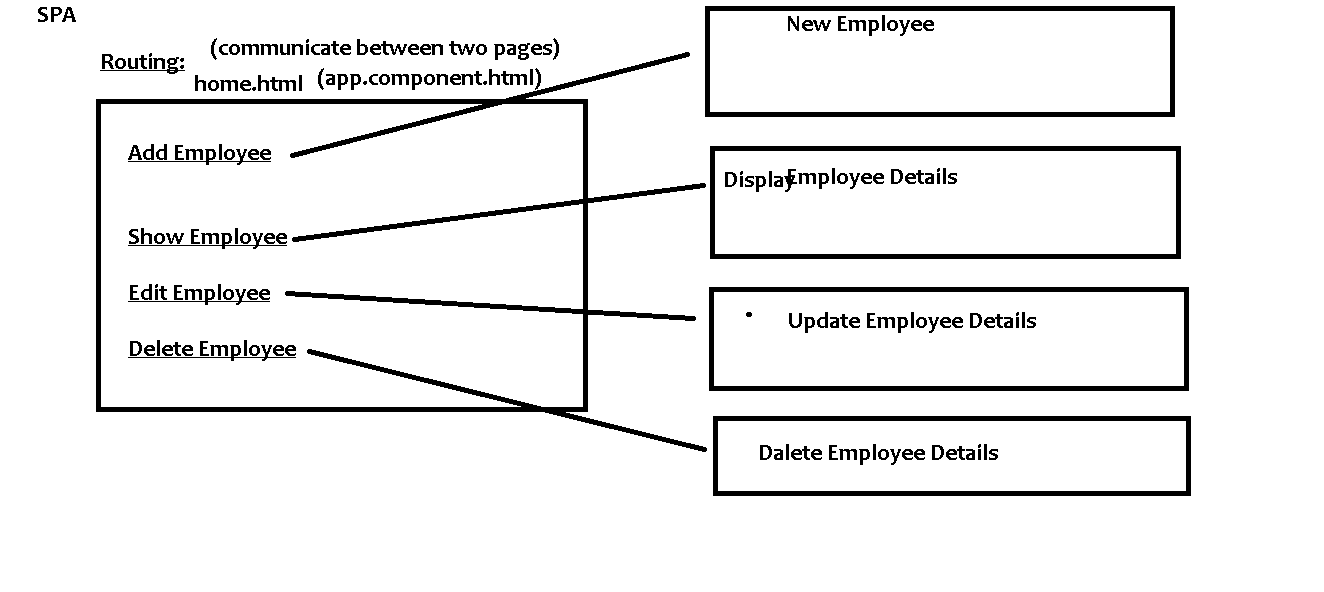
**app.component.html**

|  |
| --- |
| <div style="text-align:center">    Uppercase Pipe: {{title | uppercase}} <br>  Lowercase Pipe: {{title | lowercase}} <br>  Date in formate: {{todaydate | date:'d/M/y'}}<br/>  Date in shortTime: {{todaydate | date:'shortTime'}}<br/>  Currency Pipe: {{6589.23 | currency:"USD"}}<br/>  Json Pipe: {{ jsonval | json }}<br/>  Percent Pipe: {{00.54565 | percent}}<br/>  Decimal Pipe: {{ 454.78787814 | number: '3.4' }}<br/>  Slice Pipe: {{country | slice:2:4}}    </div> |



Continuation ……

**Routing**



**Step-1:** Create link in app.component.html

<a routerLink="add">Add Employee</a>

**step-2:** Create componant

C:\Users\t-Ravindra1\Angular6App>ng g c employee/addEmployee --spec=false --flat=true

It will be created in app.module.ts file.

import { AddEmployeeComponent } from './employee/add-employee.component';

**Step-3:**Import router module in app.module.ts

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

**step-4:** Write the path and component name in app.module.ts

RouterModule.forRoot([

{

path: 'add',

component: AddEmployeeComponent

}

])

**Step-5:**Write the Type script code in add-employee.component.ts

export class AddEmployeeComponent implements OnInit {

employeeId=1122;

constructor() { }

ngOnInit() {

}

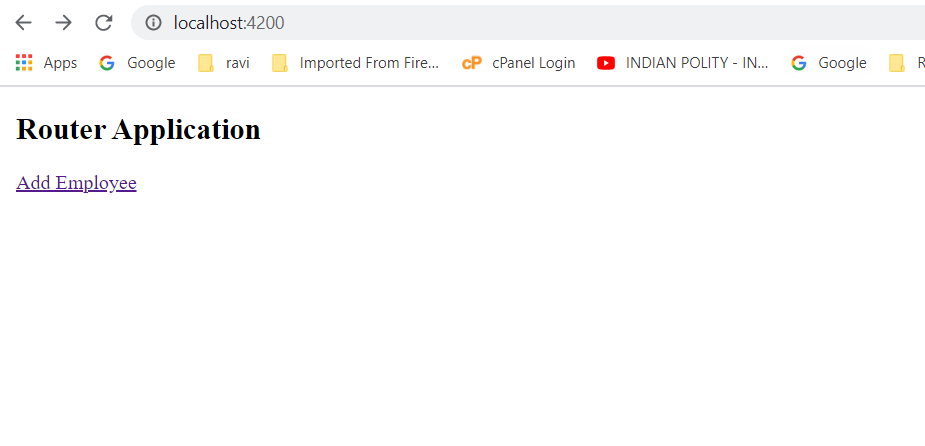
}

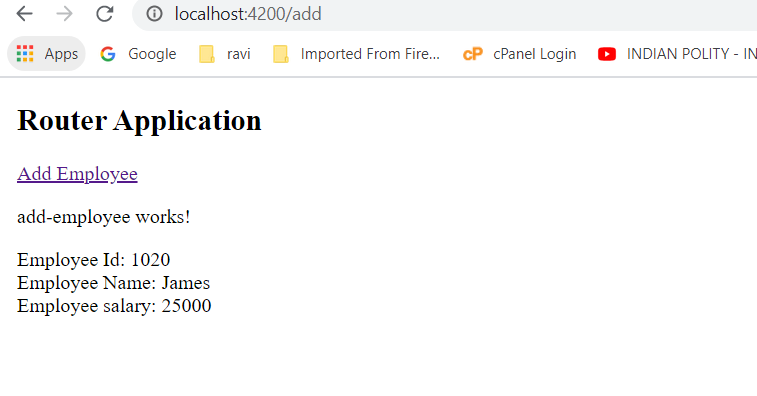
**Step-6:**Display the Result in add-employee.component.html

Employee Id: {{employeeId}}

**Step-7:** output

<http://localhost:4200/>





**Example:**

**Step-1:**

|  |
| --- |
| <!--The content below is only a placeholder and can be replaced.-->  <div style="text-align:center">    **<a routerLink="add">Add Movie</a>**    </div>  <router-outlet></router-outlet> |

**Step-2:**

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  @Component({  selector: 'app-add-employee',  templateUrl: './add-employee.component.html',  styleUrls: ['./add-employee.component.css']  })  export class AddEmployeeComponent implements OnInit {  **employeeId=1122;**  constructor() { }  ngOnInit() {  }  } |

**Step-3:**

|  |
| --- |
| <p>Employee Details</p>  <p>Employee Id: {{employeeId}}</p> |

**Step-4:**

|  |
| --- |
| import { BrowserModule } from '@angular/platform-browser';  import { NgModule } from '@angular/core';  import { RouterModule} from '@angular/router';  import { AppRoutingModule } from './app-routing.module';  import { AppComponent } from './app.component';  import { CreateNewComponentComponent } from './NewCmpComponent/create-new-component.component';  import { NewCmpComponent } from './NewCmpComponent/new-cmp.component';  **import { AddEmployeeComponent } from './employee/add-employee.component';**  import { AddComponent } from './movie/add.component';  import { MyserviceService } from './myservice.service';  @NgModule({  declarations: [  AppComponent,  CreateNewComponentComponent,  NewCmpComponent,  **AddEmployeeComponent,**  AddComponent,    ],  imports: [  BrowserModule,  AppRoutingModule,  **RouterModule.forRoot([**  **{**  **path: 'add',**  **component: AddComponent**  **}**  **])**    ],    providers: [MyserviceService],  bootstrap: [AppComponent]  })  export class AppModule { } |

**Services:**

It can be for data connection that needs to be shared across components, etc. Services help us achieve that. With services, we can access methods and properties across other components in the entire project.

Step-1:

|  |
| --- |
| C:\Users\t-Ravindra1\Angular6App>**ng g service myservice**  CREATE src/app/myservice.service.spec.ts (348 bytes)  CREATE src/app/myservice.service.ts (138 bytes) |

Step-2: myservice.service (automatically created after step-1)

|  |
| --- |
| import { Injectable } from '@angular/core';  @Injectable()  export class MyserviceService {  serviceproperty = "Service Created";  constructor() { }  showTodayDate() {  let ndate = new Date();  return ndate;  }  } |

**Step-3: app.module**

|  |
| --- |
| import { BrowserModule } from '@angular/platform-browser';  import { NgModule } from '@angular/core';  import { RouterModule} from '@angular/router';  import { AppRoutingModule } from './app-routing.module';  import { AppComponent } from './app.component';  import { CreateNewComponentComponent } from './NewCmpComponent/create-new-component.component';  import { NewCmpComponent } from './NewCmpComponent/new-cmp.component';  import { AddEmployeeComponent } from './employee/add-employee.component';  import { AddComponent } from './movie/add.component';  **import { MyserviceService } from './myservice.service';**  @NgModule({  declarations: [  AppComponent,  CreateNewComponentComponent,  NewCmpComponent,  AddEmployeeComponent,  AddComponent,    ],  imports: [  BrowserModule,  AppRoutingModule,  RouterModule.forRoot([  {  path: 'add',  component: AddComponent  }  ])    ],    **providers: [MyserviceService],**  bootstrap: [AppComponent]  })  export class AppModule { } |

**Step-4: app.component.ts**

|  |
| --- |
| import { Component } from '@angular/core';  import { MyserviceService } from './myservice.service';  @Component({  selector: 'app-root',  templateUrl: './app.component.html',  styleUrls: ['./app.component.css']  })  export class AppComponent {  title = 'Angular 6 Project!';  todaydate;  componentproperty;  constructor(private myservice: MyserviceService) {}  ngOnInit() {  this.todaydate = this.myservice.showTodayDate();  console.log(this.myservice.serviceproperty);  this.myservice.serviceproperty = "component created"; // value is changed.  this.componentproperty = this.myservice.serviceproperty;  }    } |

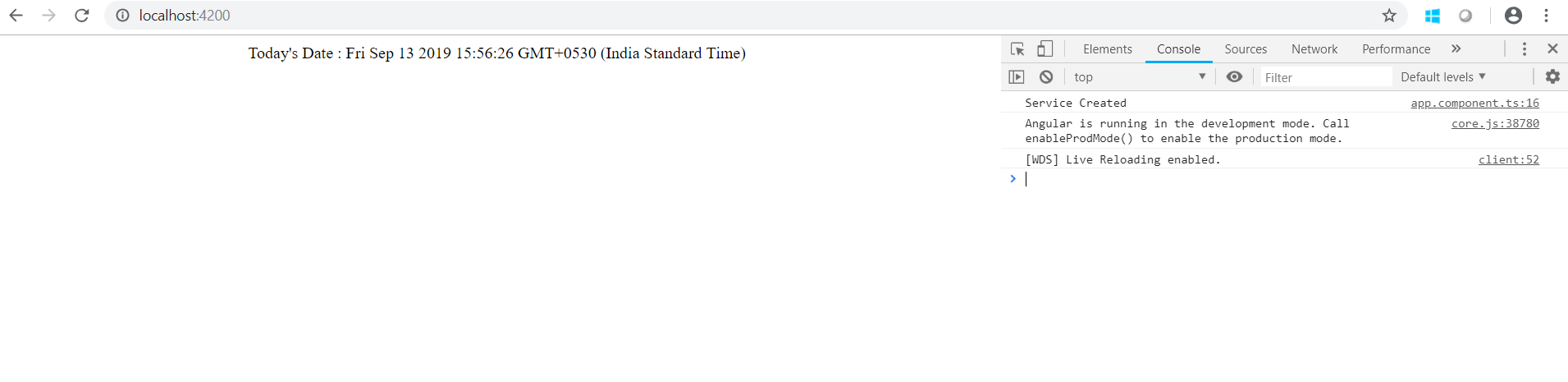
**Step-5: new-cmp.component.ts**

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  import { MyserviceService } from '../myservice.service';  @Component({  selector: 'app-new-cmp',  templateUrl: './new-cmp.component.html',  styleUrls: ['./new-cmp.component.css']  })  export class NewCmpComponent implements OnInit {  todaydate;  newcomponentproperty;  newcomponent = "Entered in newcomponent";  constructor(private myservice: MyserviceService) {}  ngOnInit() {  this.todaydate = this.myservice.showTodayDate();  this.newcomponentproperty = this.myservice.serviceproperty;  }  } |

**Step-6: app.component.html**

|  |
| --- |
| <!--The content below is only a placeholder and can be replaced.-->  <div style="text-align:center">  Today's Date : {{todaydate}}  </div>  <router-outlet></router-outlet> |

**OUTPUT:**



**Forms:**

Angular forms are used to handle user's input.

**Types:**

1. Reactive forms

2. Template-driven forms

**1. Reactive Forms:**

Reactive forms are more robust.

Reactive forms are more scalable, reusable, and testable.

**Example:**

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

export class TestComponent {

propertyname = new FormControl('');

}

**2. Template-driven forms:**

Template-driven forms are best if you want to add a simple form to your application like email list signup form.

**Example:**

export class TestComponent {

propertyname = '';

}

**Example: Angular Reactive Forms**

**step-1: Register the reactive forms module in app.module.ts file**

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

Add in imports

imports: [

ReactiveFormsModule

],

})

**step 2: Generate new component**

ng generate component RegistrationForm

**step-3: Import FormControl from @angular/forms in registration-form.component.ts**

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

and

write the following code in class:

username = new FormControl('');

**step-4: Write the following code in registration-form.component.html**

User Name: <input type="text" [formControl]="username">

**step-5: Write the following tag in app.component.html**

<app-registration-form></app-registration-form>

**step-6: compile the application using following command**

ng serve

**step-7: Run the application on browser using following url**

http://localhost:4200/

