**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:**

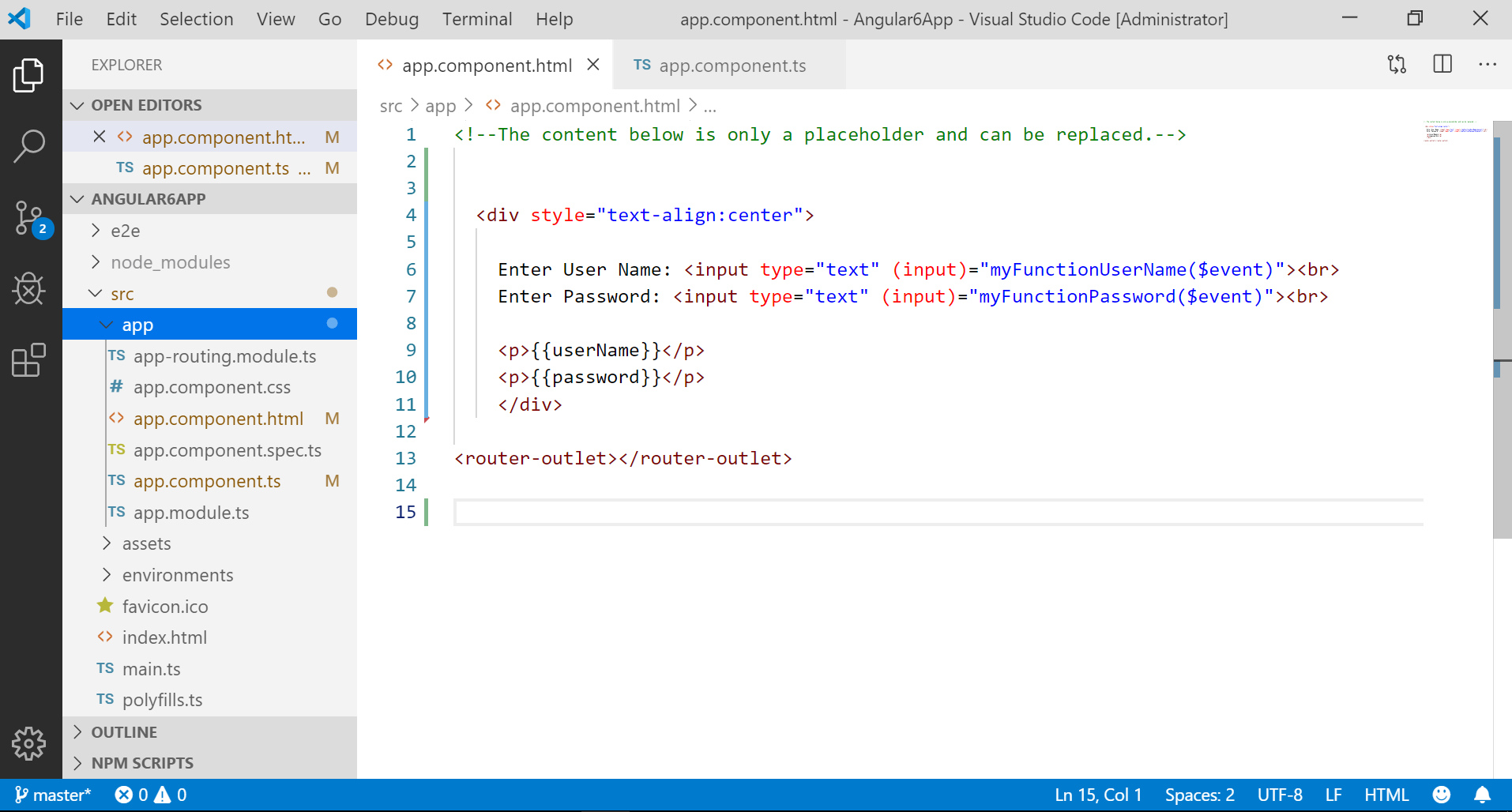
employee.component.html

employee.component.spec.ts

employee.component.ts

employee.component.css

**Derectory Structure in Visual Studio:**



**Topic: Data Binding:**

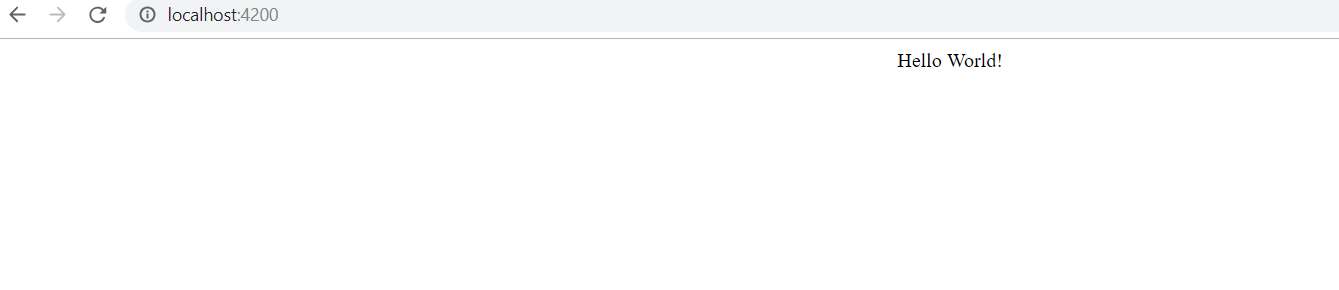
**Example-1:**

**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> |

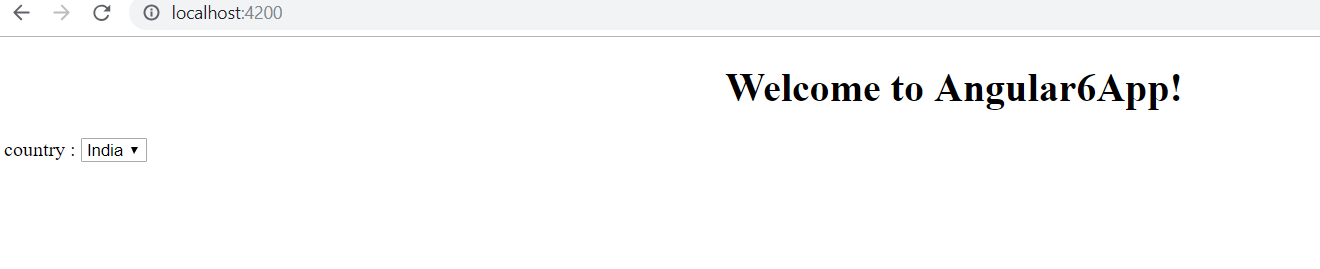
**Example-3:**

**app.component.ts**

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

**app.component.html**

|  |
| --- |
| <div> country :  <select>  <option \*ngFor = "let i of country">{{i}}</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> |

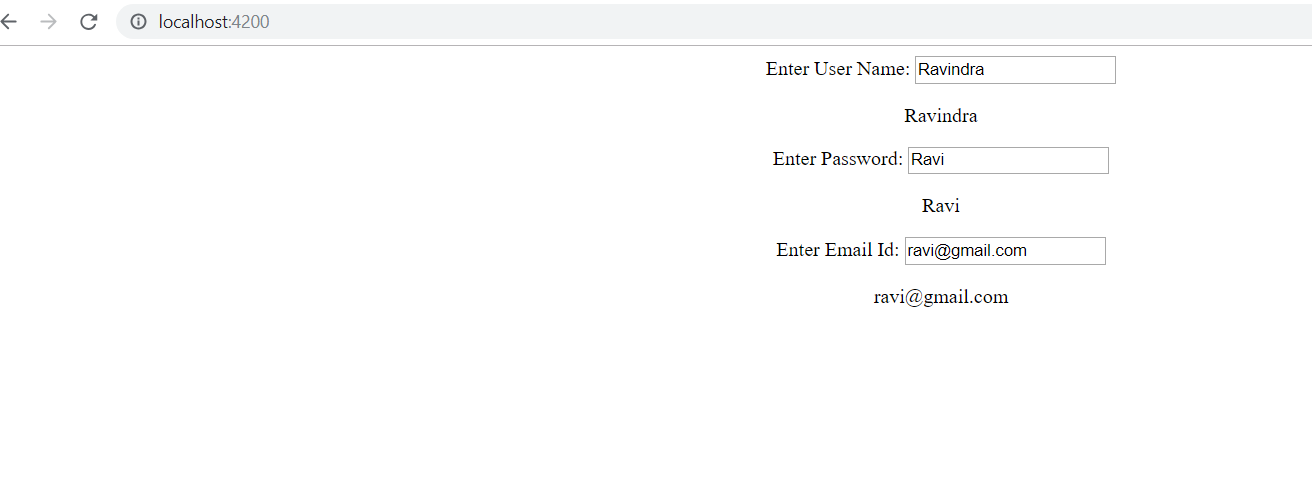
**Example-2: username,password etc..**

**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.

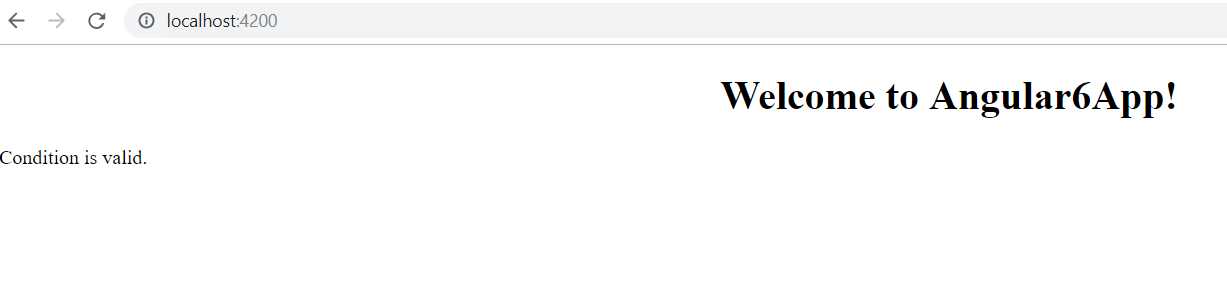
**Condition Validation (If)**

**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 {  title = 'Angular6App';  country = ["India", "USA", "UK"];  isavailable = true; //valid  } |

**app.component.html**

|  |
| --- |
| <div>  <span \*ngIf = "isavailable; then condition1 else condition2">Condition is valid.</span>  <ng-template #condition1>Condition is valid</ng-template>  <ng-template #condition2>Condition is invalid</ng-template>  </div> |



**Constructor:**

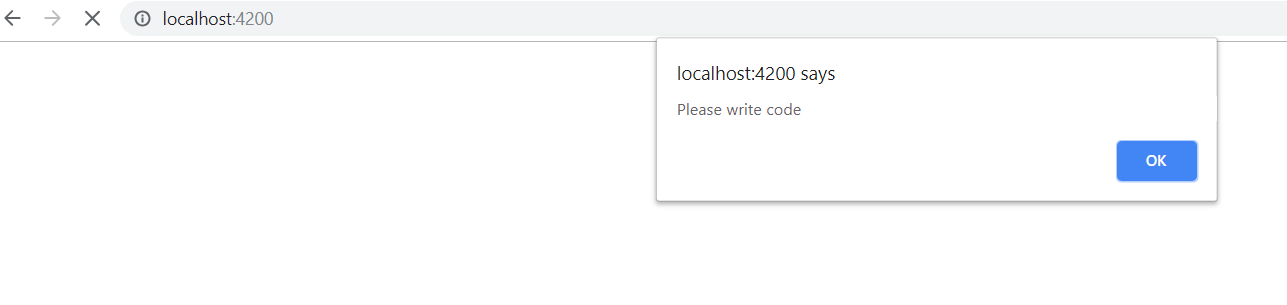
**app.component.ts**

|  |
| --- |
| 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

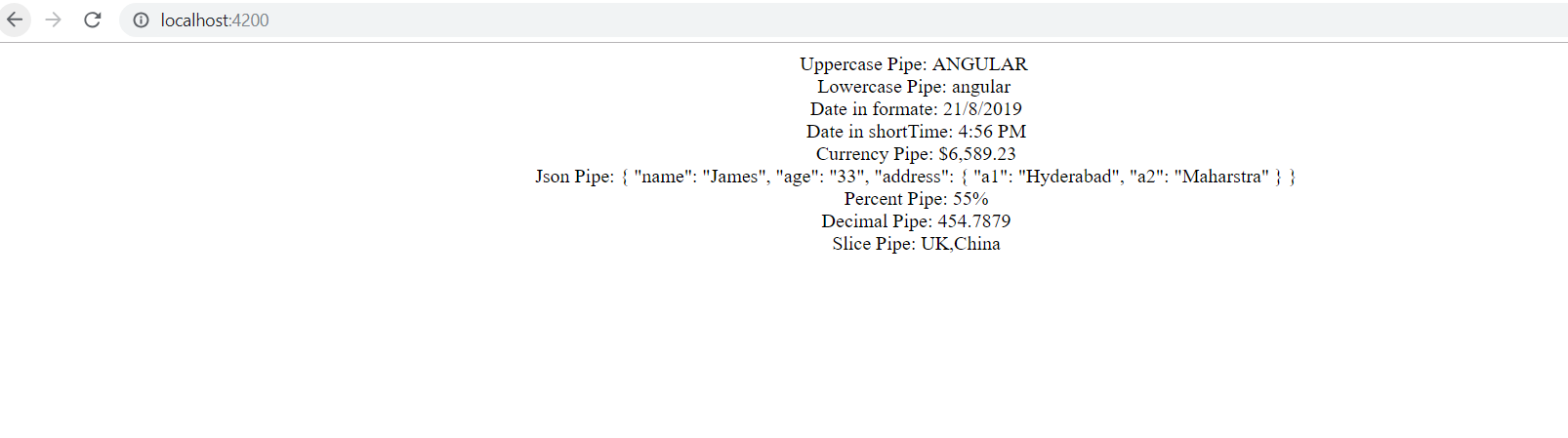
**Example-2:**

**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 ……