

ANGULAR 7

by Harsha Vardhan (UI Expert)



Contents

| | |
|---|-----------|
| Angular 7 Tutorial..... | 8 |
| TypeScript 2 | 8 |
| Introduction to TypeScript | 8 |
| Advantages of TypeScript..... | 9 |
| Steps to Prepare First Example in TypeScript | 9 |
| 1) Installing NodeJS..... | 10 |
| 2) Installing TypeScript | 14 |
| 3) Create Application Folder | 15 |
| 4) Installing Visual Studio Code | 17 |
| 5) Create TypeScript Program..... | 22 |
| 6) Compile the TypeScript Program | 24 |
| 7) Execute the TypeScript Program..... | 25 |
| TypeScript Basics | 26 |
| Variables | 26 |
| Data Types..... | 26 |
| Variables and Data Types - Example..... | 26 |
| TypeScript OOP..... | 27 |
| Object..... | 27 |
| Objects - Example..... | 28 |
| Class | 29 |
| Classes - Example..... | 30 |
| Constructor | 31 |
| Constructor - Example..... | 32 |
| Inheritance | 33 |
| Inheritance - Example..... | 35 |
| Access Modifiers | 36 |
| Access Modifiers - Example | 37 |
| Interfaces..... | 39 |
| Interfaces - Example..... | 40 |
| Enumerations | 41 |
| Enumerations - Example | 42 |
| Modules | 43 |
| Modules - Example | 43 |
| Angular 7 | 46 |
| Introduction to Angular 7 | 46 |
| What is Angular? | 46 |
| Goals of Angular | 46 |

| | |
|---|-----------|
| Versions of Angular | 47 |
| Angular (vs) AngularJS | 47 |
| Angular (vs) jQuery | 48 |
| Angular (vs) React | 48 |
| Features of Angular | 48 |
| Browser Compatibility of Angular 2+ | 48 |
| Fundamentals of Angular | 49 |
| Building Blocks of Angular | 49 |
| Angular Architecture | 49 |
| Types of compilation in Angular | 49 |
| Steps to Prepare First Application in Angular 2+ | 50 |
| 1) Installing NodeJS | 50 |
| 2) Installing TypeScript | 55 |
| 3) Installing Visual Studio Code | 56 |
| 4) Create Application Folder | 61 |
| 5) Install @angular/cli package | 63 |
| 6) Creating New Angular Application | 64 |
| 7) Open the Application in Visual Studio Code | 65 |
| 8) app.component.html | 67 |
| 9) Compile the application | 68 |
| 10) Run the application | 69 |
| Folder Structure of Angular Application | 69 |
| 1) package.json | 70 |
| Packages of Angular 2+ | 74 |
| 2) tsconfig.json | 77 |
| 3) tslint.json | 79 |
| 4) protractor.conf.js | 82 |
| 5) karma.conf.js | 83 |
| 6) .angular-cli.json | 84 |
| 7) polyfills.ts | 86 |
| 8) src/styles.css | 87 |
| 9) src/index.html | 88 |
| 10) src/main.ts | 88 |
| 11) src/app/app.module.ts | 89 |
| 12) src/app/app.component.ts | 89 |
| 13) src/app/app.component.html | 90 |
| 14) src/app/app.component.css | 90 |
| 15) src/app/app.component.spec.ts | 90 |
| Components | 91 |
| Modules | 92 |

| | |
|--|-----|
| Data Bindings | 93 |
| Data Bindings - Example | 95 |
| Login - Example..... | 99 |
| Registration Form - Example | 102 |
| Built-in Directives..... | 107 |
| Style | 107 |
| Style - Example..... | 107 |
| ngClass | 109 |
| ngClass - Example | 110 |
| ngIf | 112 |
| ngIf - Example | 113 |
| ngIf and else | 115 |
| ngIf and else - Example | 116 |
| ngSwitch | 119 |
| ngSwitch - Example..... | 119 |
| ngFor..... | 122 |
| ngFor | 122 |
| ngFor - Example | 122 |
| ngFor with Object Array..... | 125 |
| ngFor with Object Array - Example..... | 125 |
| ngFor with Add, Remove..... | 129 |
| ngFor with Add, Remove - Example..... | 129 |
| ngFor with Searching and Sorting..... | 133 |
| ngFor with Searching and Sorting- Example..... | 134 |
| Multiple Components, Multiple Modules..... | 138 |
| Multiple Components..... | 138 |
| Multiple Components - Example..... | 139 |
| Multiple Modules..... | 145 |
| Multiple Modules - Example | 145 |
| Children of Components | 152 |
| Sharing Data from Parent Component to Child Component..... | 152 |
| Sharing Data from Parent Component to Child Component - Example..... | 152 |
| ViewChild..... | 156 |
| ViewChild - Example..... | 157 |
| ViewChildren..... | 162 |
| ViewChildren - Example | 163 |
| ContentChild | 169 |
| ContentChild - Example | 169 |
| ContentChildren | 174 |
| ContentChildren - Example | 175 |

| | |
|--|-----|
| Reference Names..... | 181 |
| Reference Names - Example..... | 181 |
| ElementRef..... | 187 |
| ElementRef - Example..... | 187 |
| Life Cycle Hooks | 192 |
| Life Cycle Hooks - Example..... | 194 |
| Services..... | 201 |
| Services - Example | 202 |
| Sharing Data using Services..... | 206 |
| Sharing Data using Services - Example..... | 206 |
| Custom Directives | 210 |
| Custom Directives - Example..... | 211 |
| Pipes..... | 215 |
| Pipes - Example..... | 215 |
| Custom Pipes..... | 220 |
| Custom Pipes - Example..... | 221 |
| Forms and Validations | 224 |
| Template Driven Forms..... | 224 |
| Template Driven Forms - Example..... | 225 |
| Reactive Forms (or) Model Driven Forms | 230 |
| Reactive Forms - Example..... | 231 |
| Routing | 236 |
| Routing - Example..... | 238 |
| Route Parameters..... | 243 |
| Route Parameters - Example..... | 244 |
| Child Routes | 252 |
| Child Routes - Example | 253 |
| Guards..... | 264 |
| CanActivate | 265 |
| CanActivate - Example | 266 |
| CanDeactivate..... | 282 |
| CanDeactivate - Example | 283 |
| Deployment | 291 |
| Deployment to Java | 291 |
| Deployment to .NET | 295 |
| AJAX..... | 299 |
| Execution Flow of AJAX..... | 300 |
| Advantages of AJAX..... | 300 |
| Types of AJAX Request..... | 300 |
| “@angular/common/http” package | 300 |

| | |
|--|------------|
| AJAX – Java – Simple - Example | 302 |
| AJAX – Java – Get - Example | 307 |
| AJAX – Java – Search - Example | 313 |
| AJAX – Java – Insert - Example..... | 320 |
| AJAX – Java – Update - Example..... | 326 |
| AJAX – Java – Delete - Example | 332 |
| Http Interceptors..... | 338 |
| AJAX – Java – Http Interceptors - Get - Example..... | 339 |
| RxJS..... | 346 |
| Observable and Observer | 346 |
| Observable.interval..... | 348 |
| Observable.Interval - Example | 349 |
| Observable.range | 354 |
| Observable.range - Example | 355 |
| Observable.from..... | 360 |
| Observable.from - Example | 361 |
| Observable.fromEvent..... | 366 |
| Observable.fromEvent - Example..... | 367 |
| Custom Observables | 372 |
| Custom Observables - Example | 373 |
| Custom Observables using Services - Example | 379 |
| Map..... | 384 |
| Map - Example..... | 385 |
| Filter..... | 391 |
| Filter - Example | 392 |
| Take | 398 |
| Take - Example | 399 |
| Skip | 404 |
| Skip - Example | 405 |
| AJAX with Observable - Java - Get | 410 |
| Map HTTP Request - Example | 417 |
| Cancelling HTTP Request - Example..... | 423 |
| Retrying HTTP Request - Example..... | 430 |
| Unit Testing | 437 |
| Unit Testing - Example | 438 |
| Animations | 442 |
| Animations - Example..... | 444 |
| Angular Material Design | 448 |
| Packages of Angular Material Design | 448 |
| Themes of Angular Material Design | 449 |
| MatButtonModule | 450 |

HARSHA

Angular 7 Tutorial

TYPESCRIPT 2

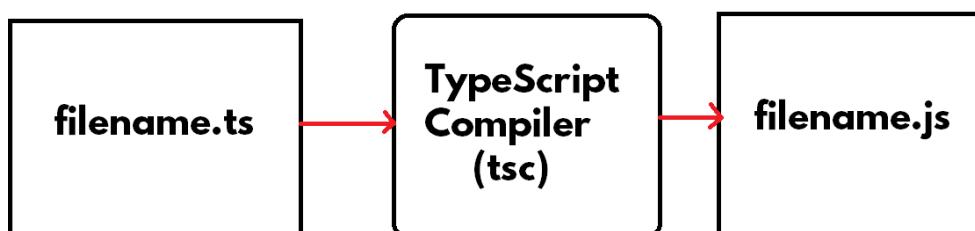
Introduction to TypeScript

What is TypeScript

- TypeScript is a programming language, which is developed based on JavaScript.
- TypeScript is a superset of JavaScript, which adds data types, classes, interfaces and other features.
- TypeScript = JavaScript + Data Types + Classes + Interfaces + Misc. Concepts (Arrow Functions + Multiline Strings + String Interpolation + Destructuring + Modules etc.)
- TypeScript is built on the top of JavaScript. That means all the code of JavaScript works as-it-is in TypeScript, but in TypeScript, we can additionally use data types, classes, interfaces etc., concepts.
- TypeScript is developed by Microsoft Corporation in 2012.

Features of TypeScript

- TypeScript is a general purpose programming language.
- TypeScript is built in the top of JavaScript. It supports all the concepts of JavaScript.
- TypeScript doesn't stick to client side programming / server side programming. TypeScript can be used in client side program development, using Angular framework. It can be used in server side program development, using NodeJS platform.
- TypeScript requires to be used in modern code editors / IDE's, such as Visual Studio Code, Atom, Sublime Text, Web Storm, Eclipse etc.
- Browser doesn't support TypeScript directly. TypeScript code can't be executed directly by the browser. So TypeScript code should be converted into JavaScript code, and we have to import JavaScript language file into the web page. Browser executes JavaScript. We use "TypeScript Compiler" (tsc) to compile / transpile "filename.ts (TypeScript file)" to "filename.js (JavaScript file)". We won't load TypeScript file into the browser; We will load & execute JavaScript file into the browser.



Versions of TypeScript

- TypeScript 0.8 : 2012 (first version)
- TypeScript 0.9 : 2013
- TypeScript 1.0 : 2014
- TypeScript 2.0 : 2016
- TypeScript 2.6 : 2017
- TypeScript 2.8 : 2018

Advantages of TypeScript

1. Static Typing and Type Safety

- **Static Typing:** Whenever we can fix a data type for the variable while declaration of variable, and we can't change its data type throughout the program, then it is said to be "static typing". Whenever we can't fix a data type for the variable while declaration, and the data type will be automatically taken by the runtime engine automatically at the time of program execution, and we can assign any type of value into the variable, then it is said to be "dynamic typing". C, C++, Java, C#.NET are examples of "static typing". "JavaScript", "Python" are examples of "dynamic typing". TypeScript supports "Optional Static Typing". That means you can use both "dynamic typing" and "static typing" in TypeScript.
- **Type Safety:** If we specify data type while declaring the variable and when we assign wrong type of value into the variable, the compiler shows errors.

2. Identification of Errors

- Typing mistakes and syntax errors are displayed as errors in the compilation time itself (rather than at runtime).

3. Classes and Interfaces

- TypeScript supports classes and interfaces; so it provides rich programming experience much like other languages such as Java, C#.NET etc.

4. Intellisense

- Automatic suggestions will be displayed while writing the code. For example, when we type "c" in the IDE, the list of matching names that start with "c" will be automatically displayed. This is available only in supporting code editors / IDE's such as Visual Studio Code, WebStorm, Sublime Text etc.

Steps to Prepare First Example in TypeScript

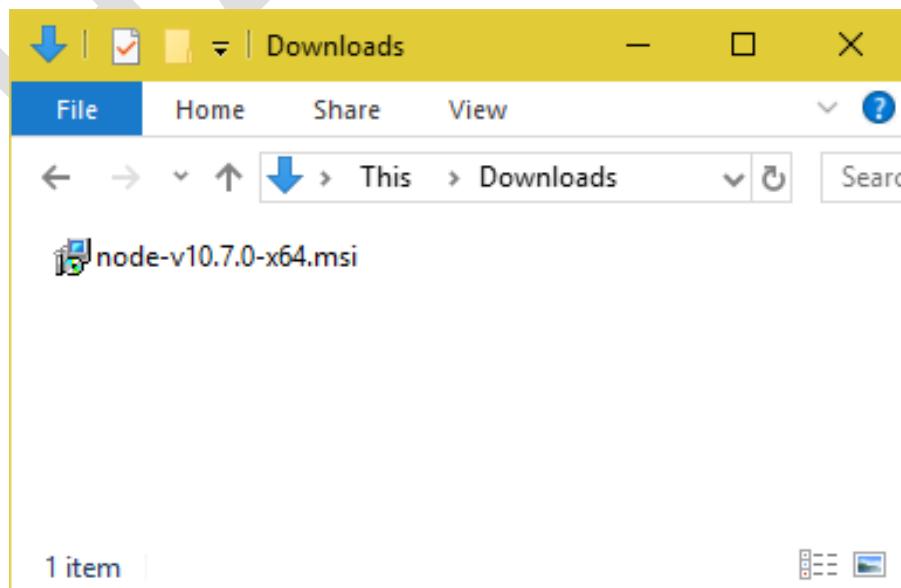
- 1) Installing NodeJS
- 2) Installing TypeScript
- 3) Create Application Folder
- 4) Installing the Code Editor: Visual Studio Code
- 5) Create TypeScript Program
- 6) Compile the TypeScript Program
- 7) Execute the TypeScript Program

1) Installing NodeJS

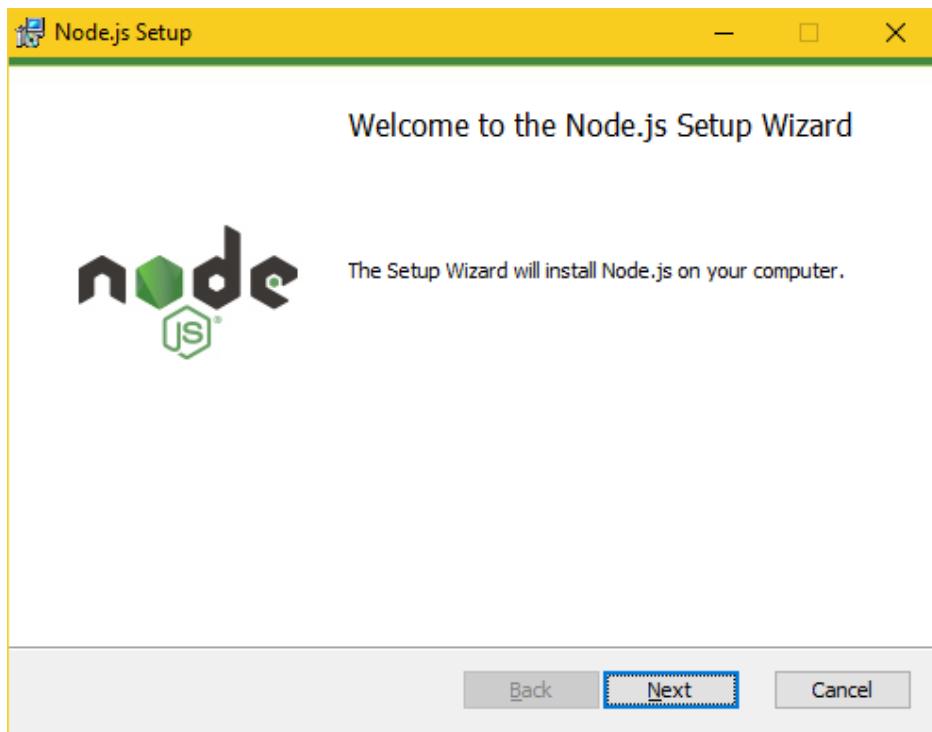
- Go to “<https://nodejs.org>”.

The screenshot shows the official Node.js website (<https://nodejs.org/en/>). At the top, there's a navigation bar with links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, NEWS, and FOUNDATION. The FOUNDATION link is highlighted with a green background. Below the navigation, a large heading says "Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine." To the right, there's an announcement for "#jsinteractive" happening on October 10-12, 2018, in Vancouver, Canada. Two large green buttons are prominently displayed: one for "8.11.3 LTS" (Recommended For Most Users) and another for "10.7.0 Current" (Latest Features). Below these buttons, there are links for "Other Downloads", "Changelog", and "API Docs". A note at the bottom suggests looking at the "Long Term Support (LTS) schedule".

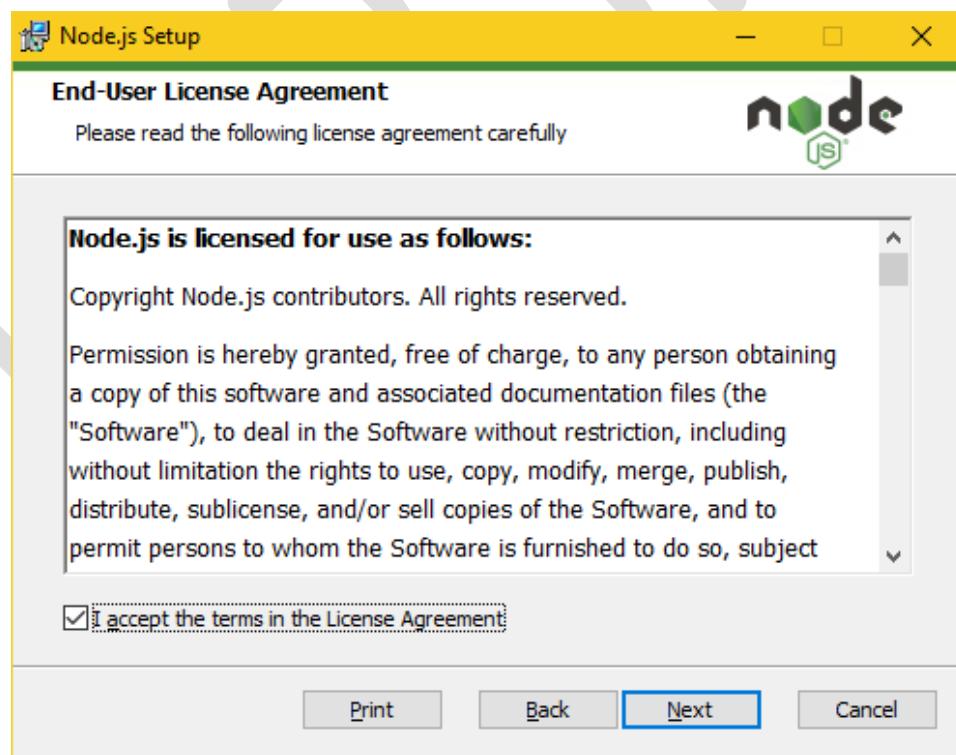
- Click on “10.7.0 Current”. **Note:** The version number can be very. Choose the latest version.
- You will download a file called “node-v10.7.0-x64.msi”.



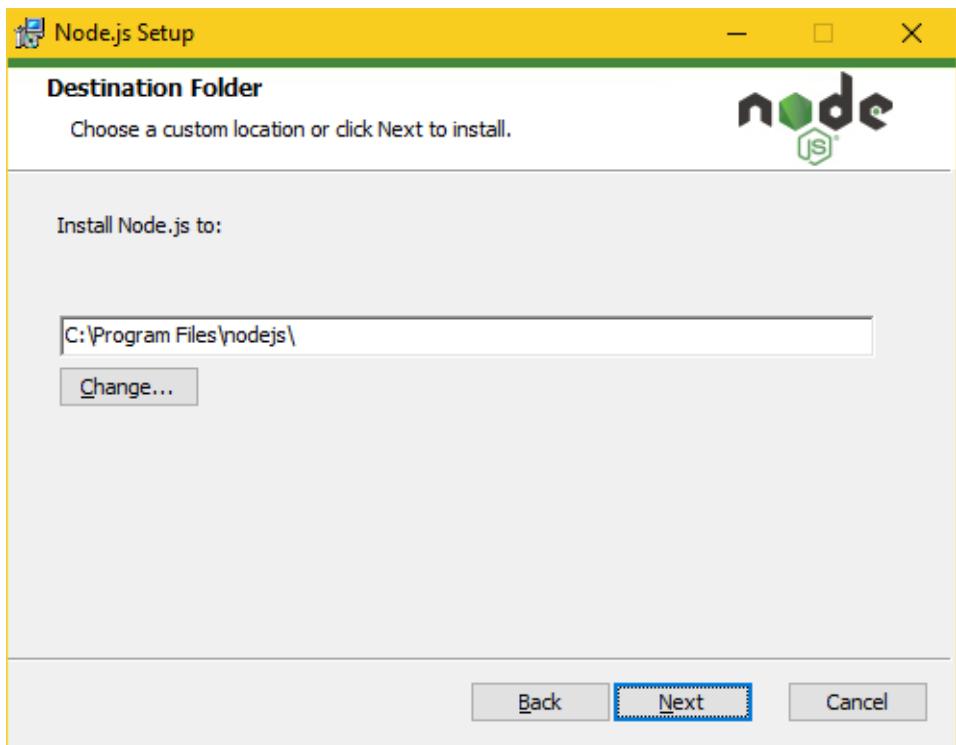
- Go to Downloads folder and double click on “node-v10.7.0-x64.msi”.



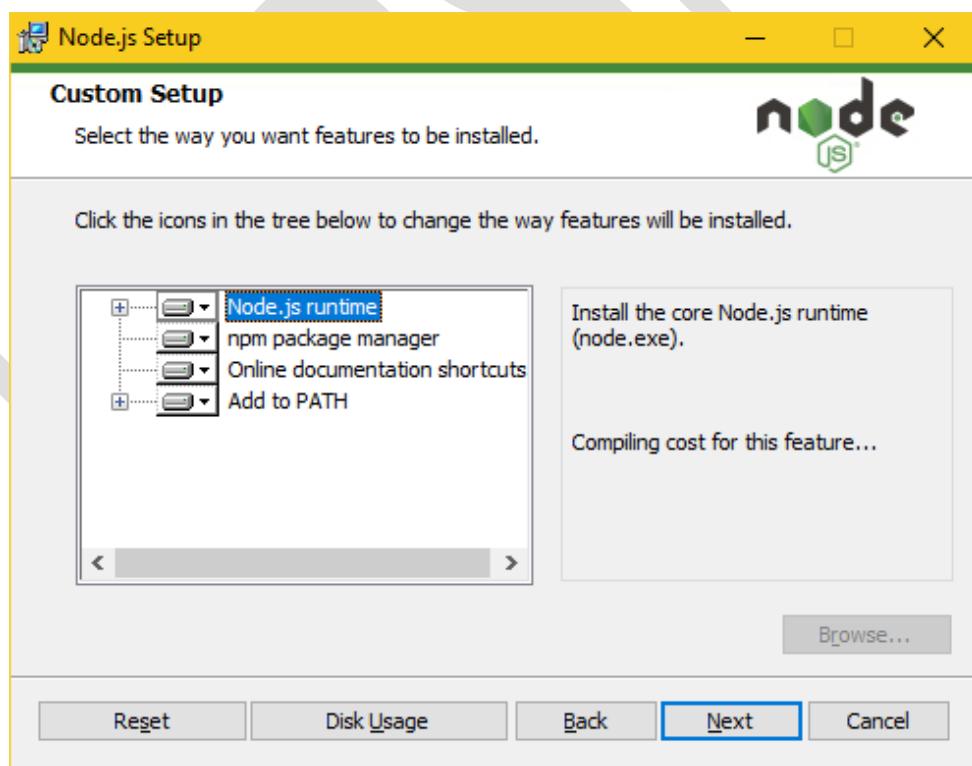
- Click on “Next”.



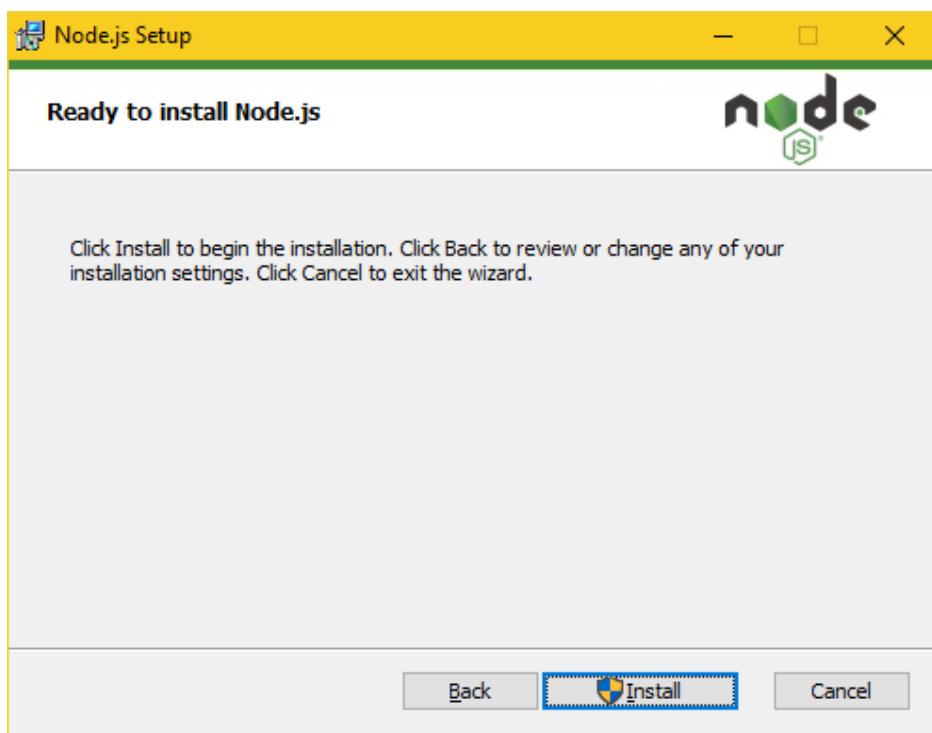
- Check the checkbox “I accept the terms in the License Agreement” and click on “Next”.



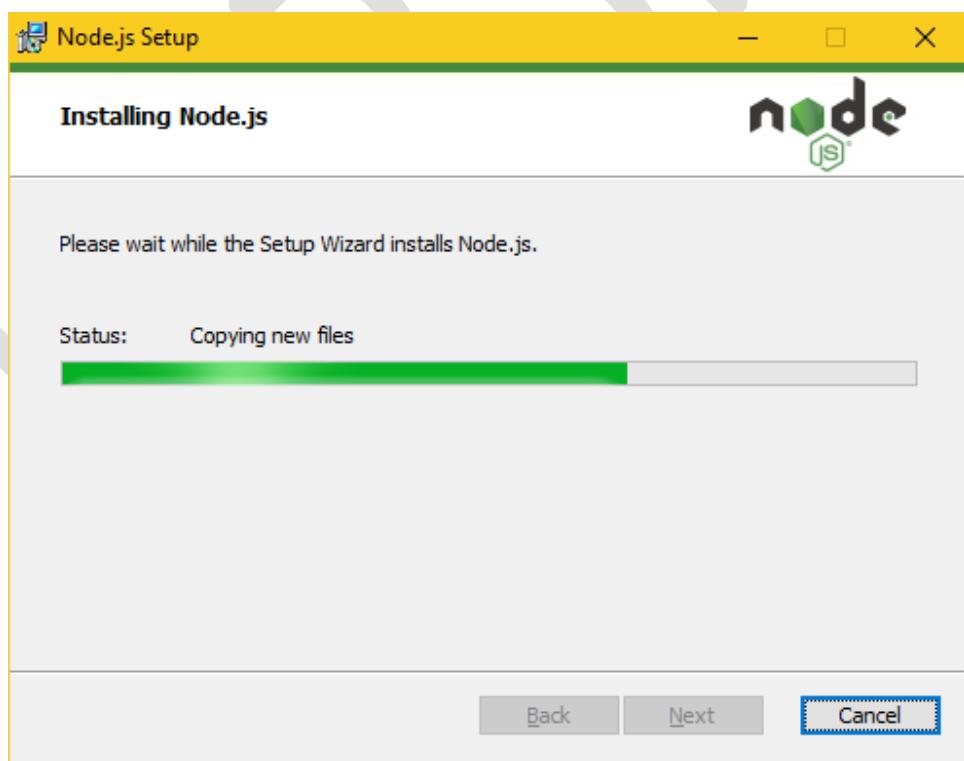
- Click on "Next".



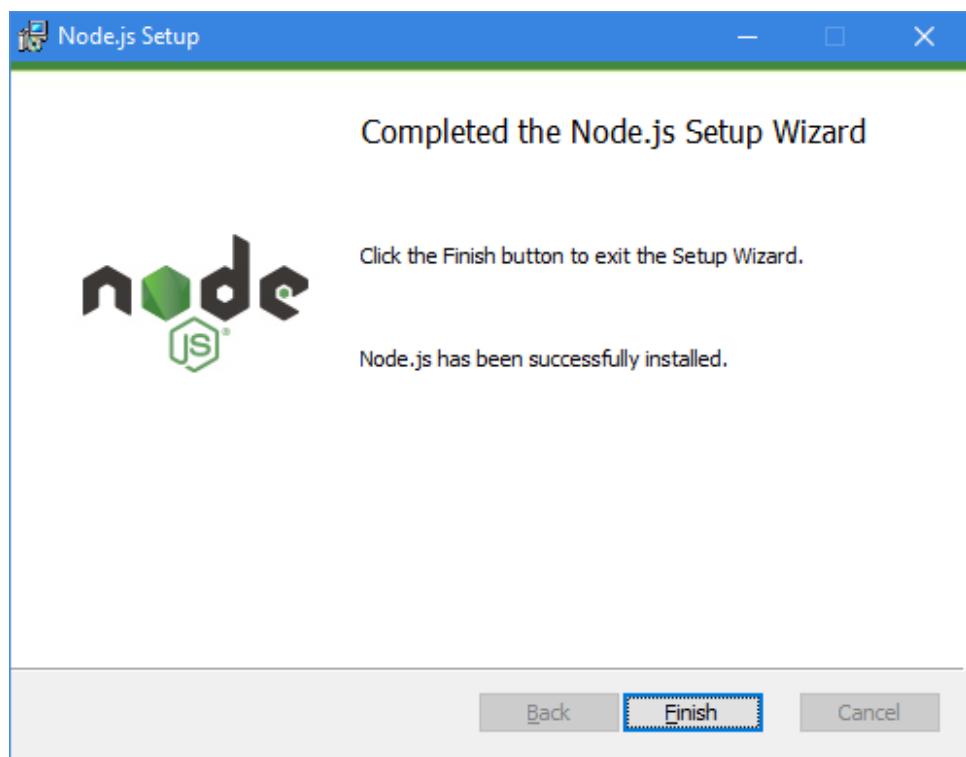
- Click on "Next".



- Click on “Install”.
- Click on “Yes”.



- Installation is in progress now.



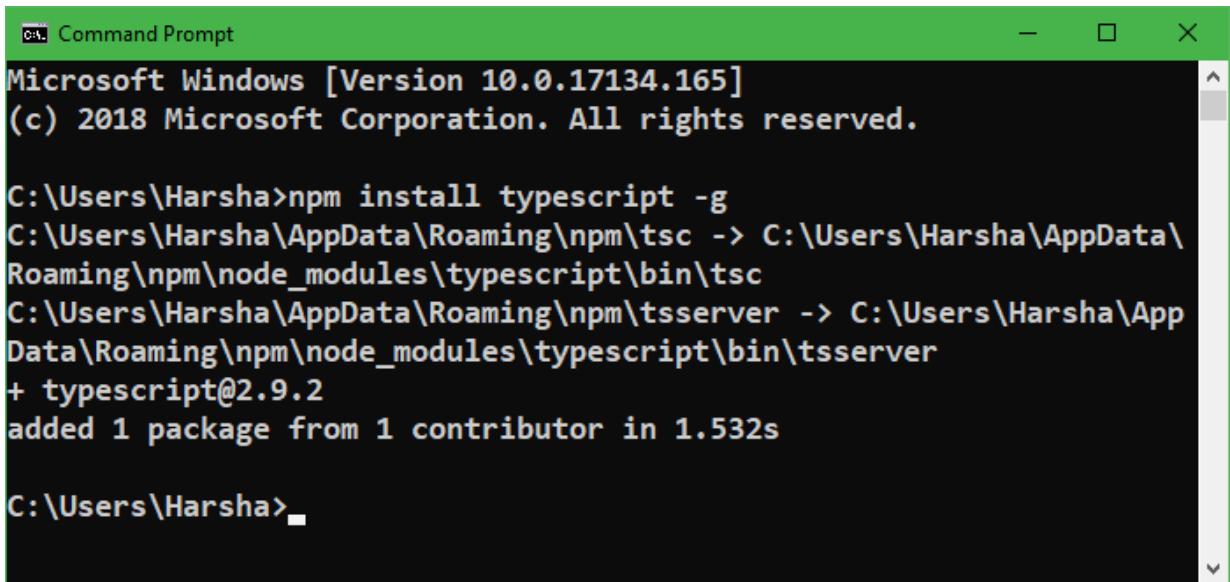
- After installation is completed, click on “Finish”. Now NodeJS installation is finished.

2) Installing TypeScript

- Open “Command Prompt”.
- Type **npm install typescript -g** in the Command prompt and press Enter.

A screenshot of a Microsoft Windows Command Prompt window. The title bar says "Command Prompt". The window shows the text "Microsoft Windows [Version 10.0.17134.165]" and "(c) 2018 Microsoft Corporation. All rights reserved.". In the command line, the user has typed "C:\Users\Harsha>npm install typescript -g" and is pressing Enter.

- After pressing Enter:



```
cmd Command Prompt
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

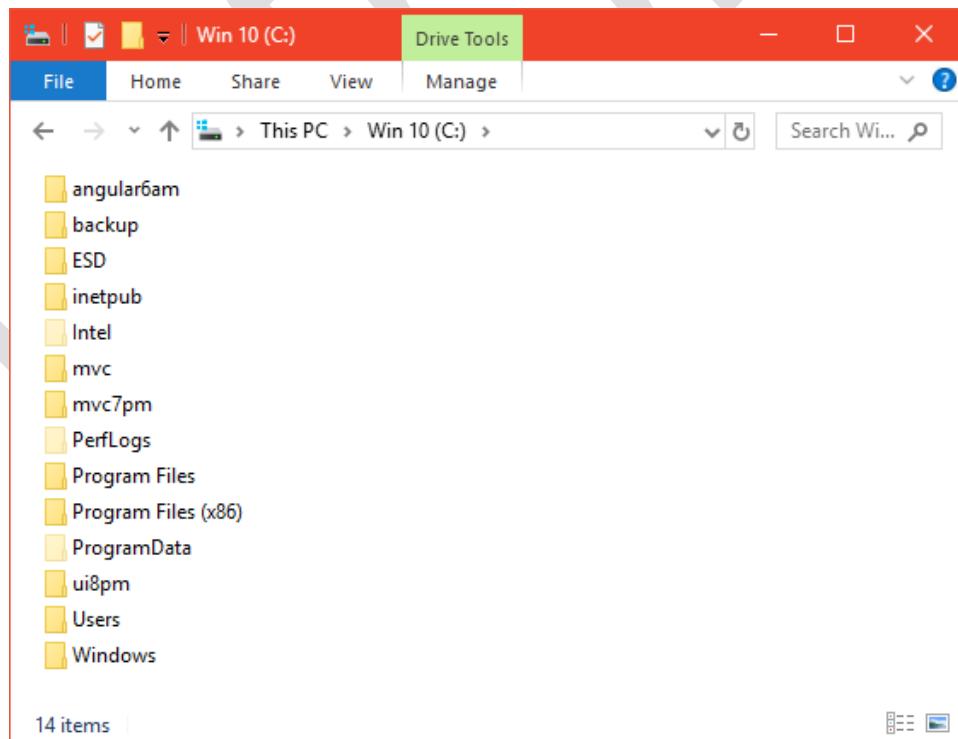
C:\Users\Harsha>npm install typescript -g
C:\Users\Harsha\AppData\Roaming\npm\tsc -> C:\Users\Harsha\AppData\Roaming\npm\node_modules\typescript\bin\tsc
C:\Users\Harsha\AppData\Roaming\npm\tsserver -> C:\Users\Harsha\AppData\Roaming\npm\node_modules\typescript\bin\tsserver
+ typescript@2.9.2
added 1 package from 1 contributor in 1.532s

C:\Users\Harsha>
```

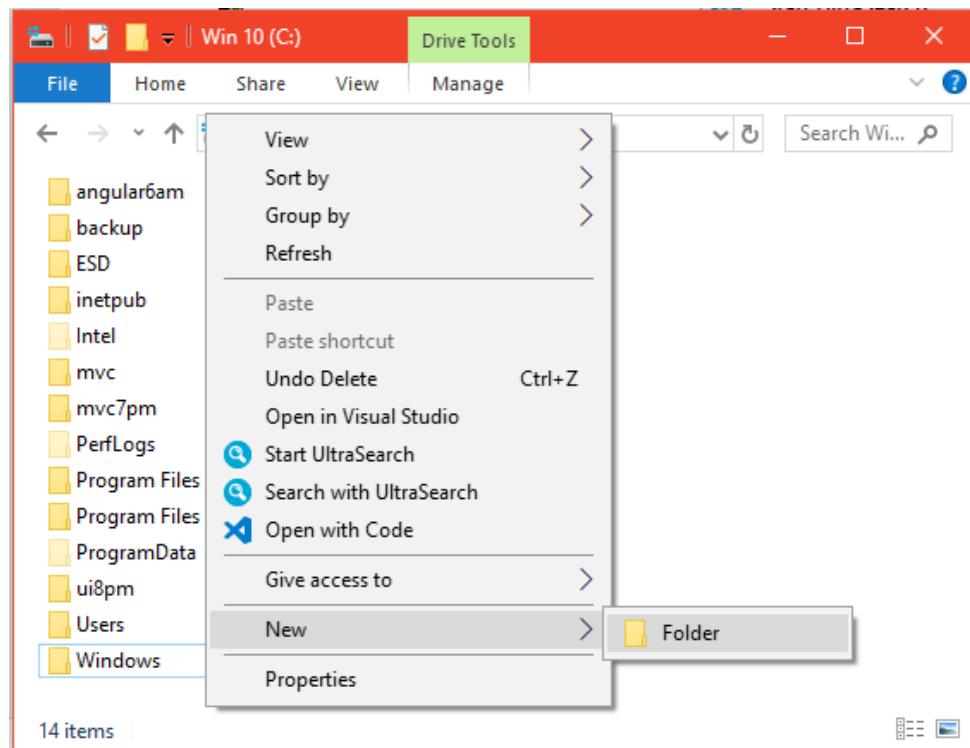
- Now TypeScript successfully installed.

3) Create Application Folder

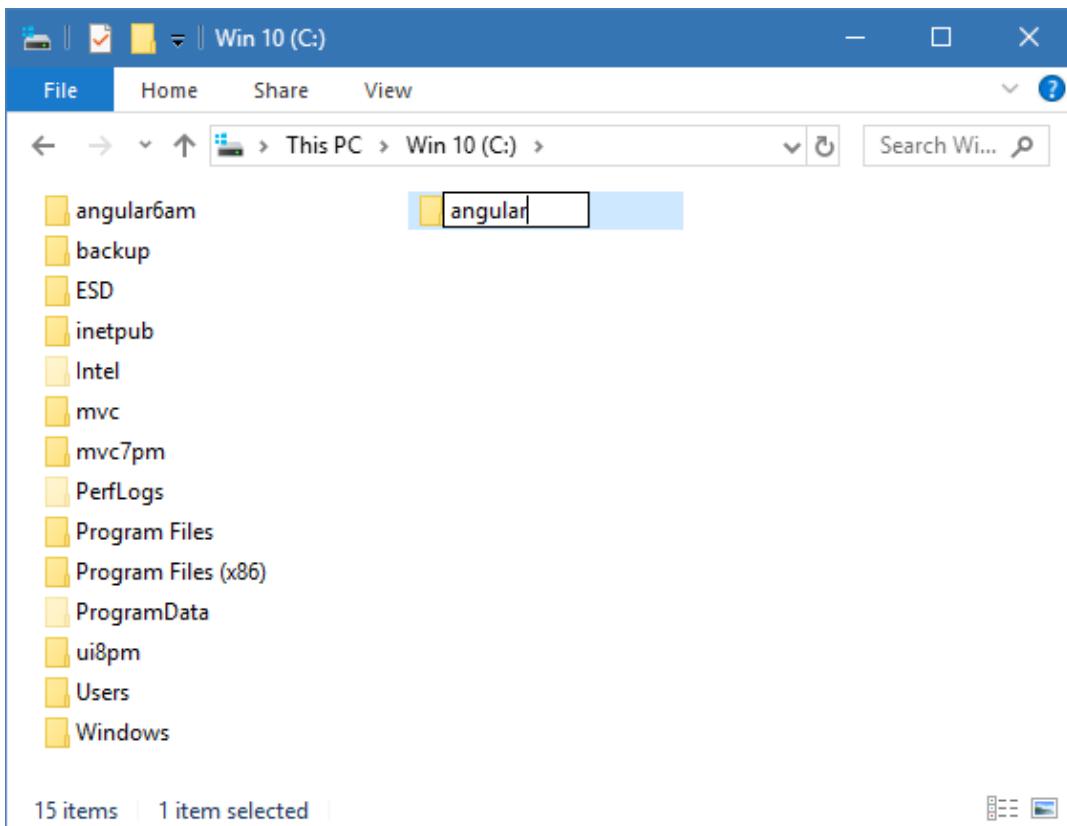
- Go to “Computer” (or) “This PC”.
- Go to “c:\”.



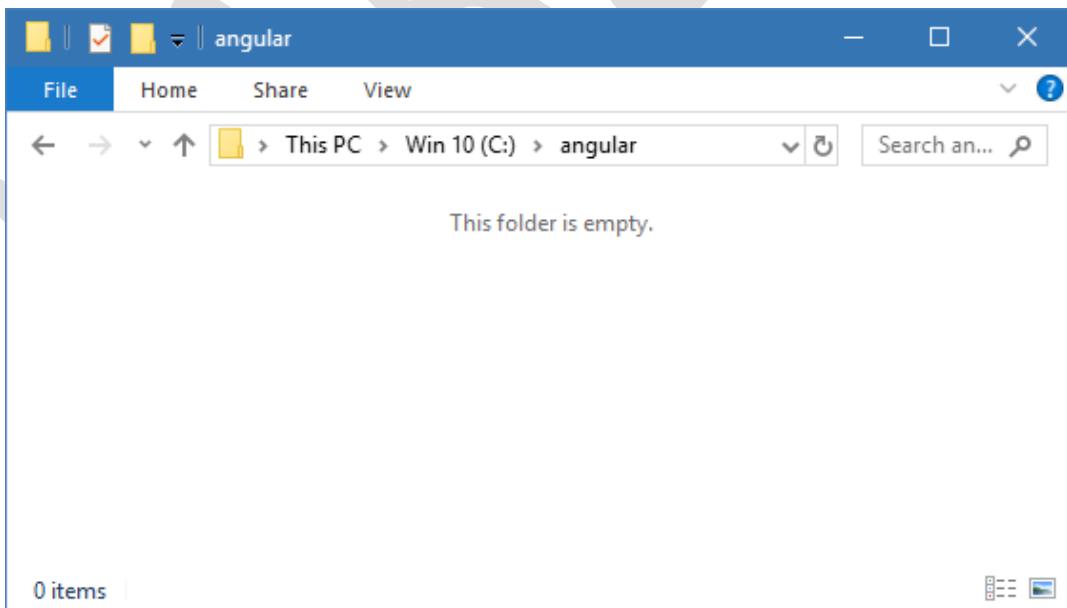
- Right click on the empty area and click on “New Folder”.



- Type the folder name as “angular” and press Enter.



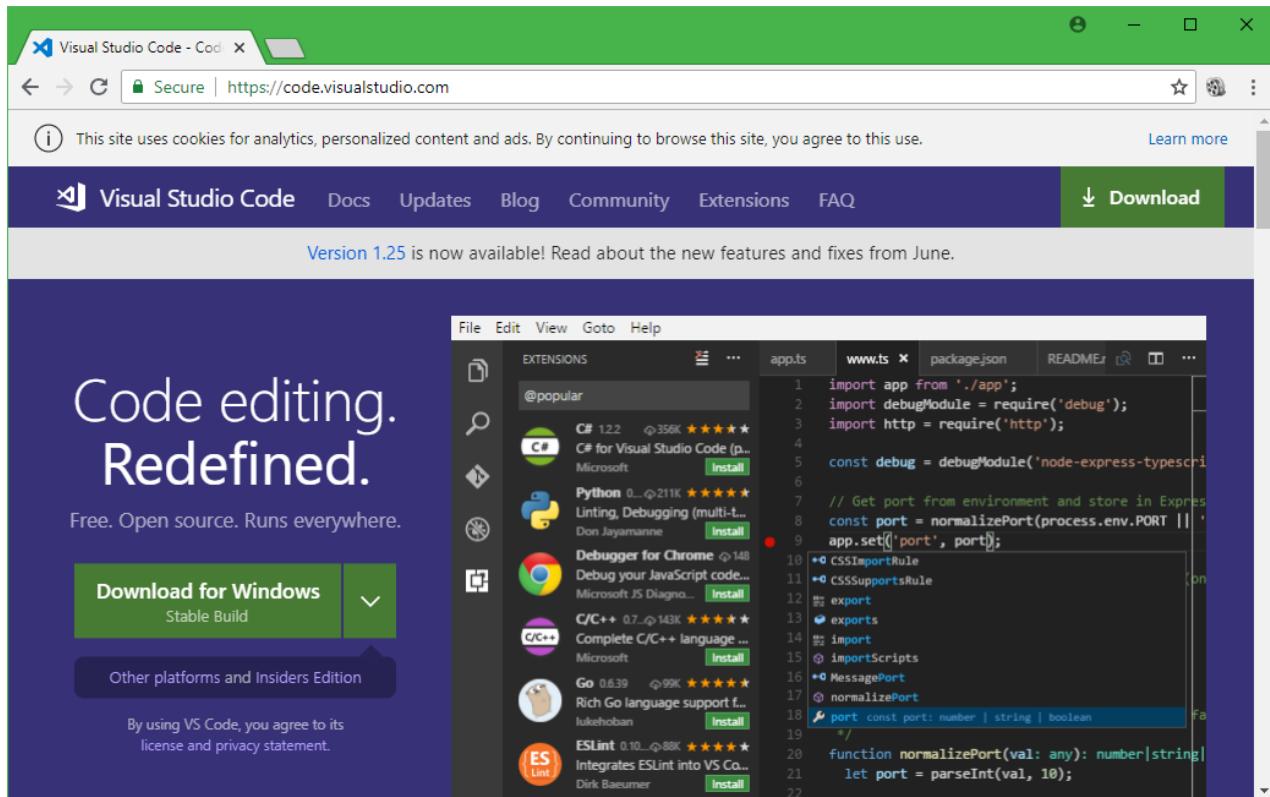
- The “c:\angular” folder is ready now.



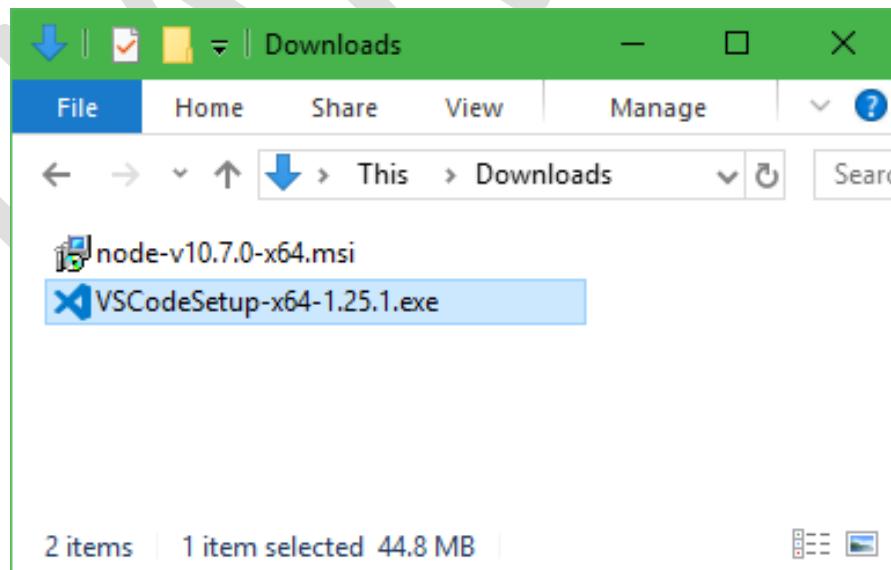
4) Installing Visual Studio Code

- “Visual Studio Code” is the recommended editor for typescript and angular.
- Visual Studio Code is easy, free, modern, customizable, open source, cross platform editor for editing so many languages such as Html, Css, JavaScript, TypeScript, Solidity, C#, Python etc.

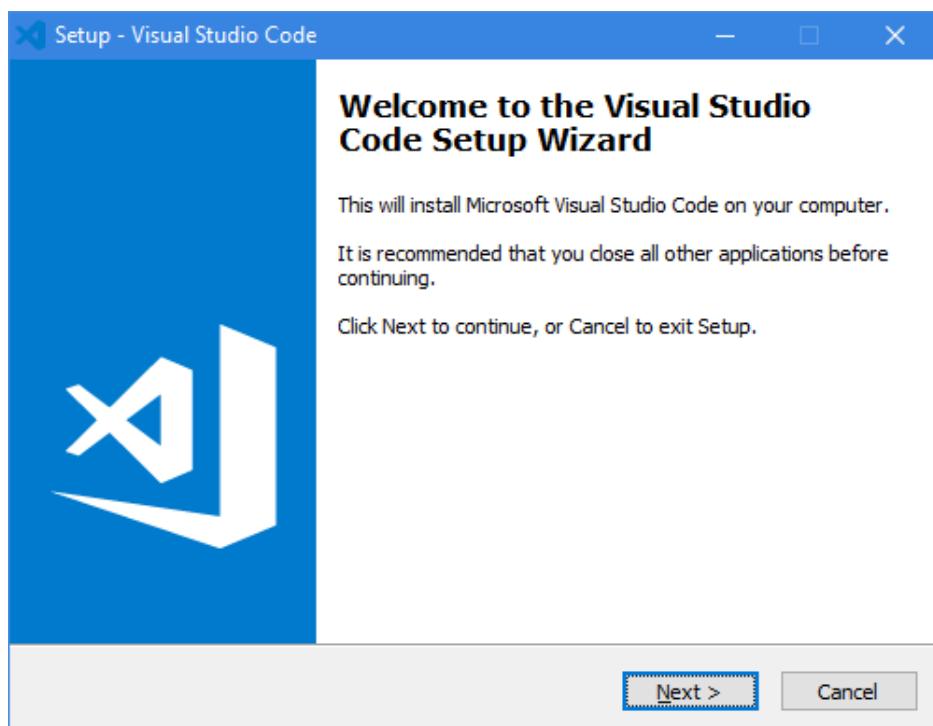
- Go to <https://code.visualstudio.com>



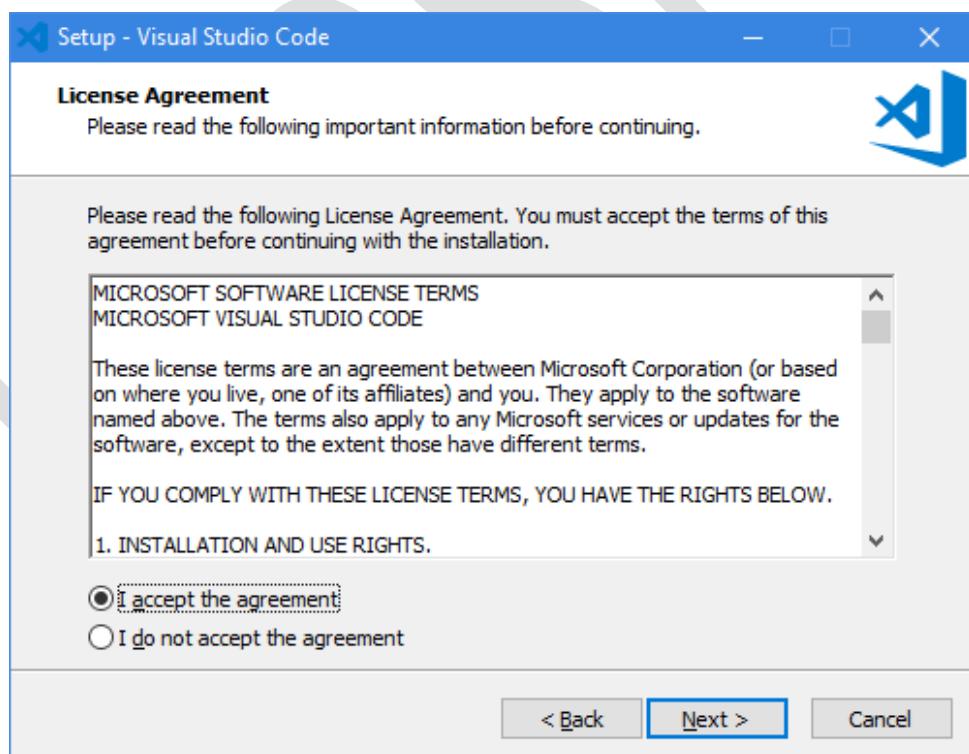
- Click on “Download for Windows”.
- **Note:** The version number may be different at your practice time.
- Go to “Downloads” folder; you can find “VSCodeSetup-x64-1.25.1.exe” file.



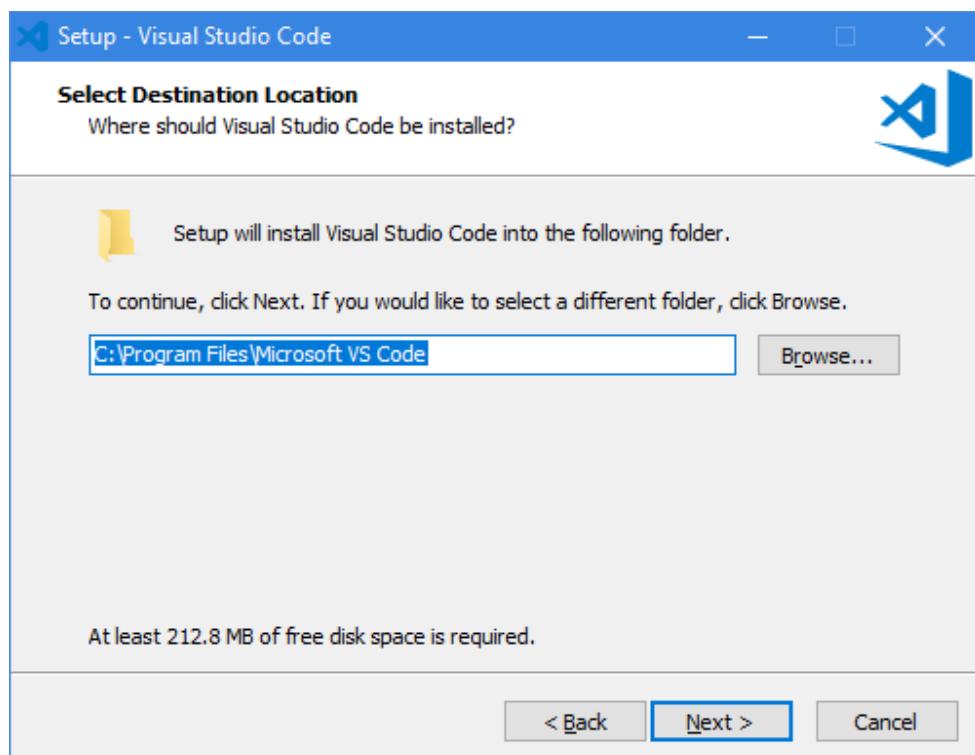
- Double click on “VSCodeSetup-x64-1.25.1.exe” file.
- Click on “Yes”.



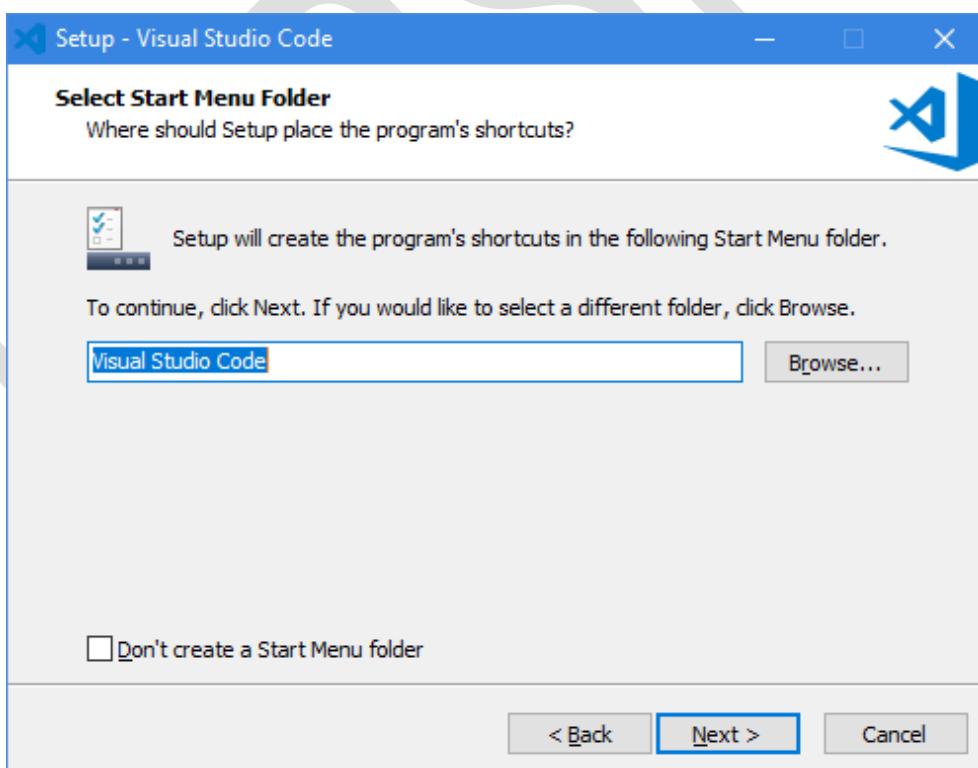
- Click on “Next”.



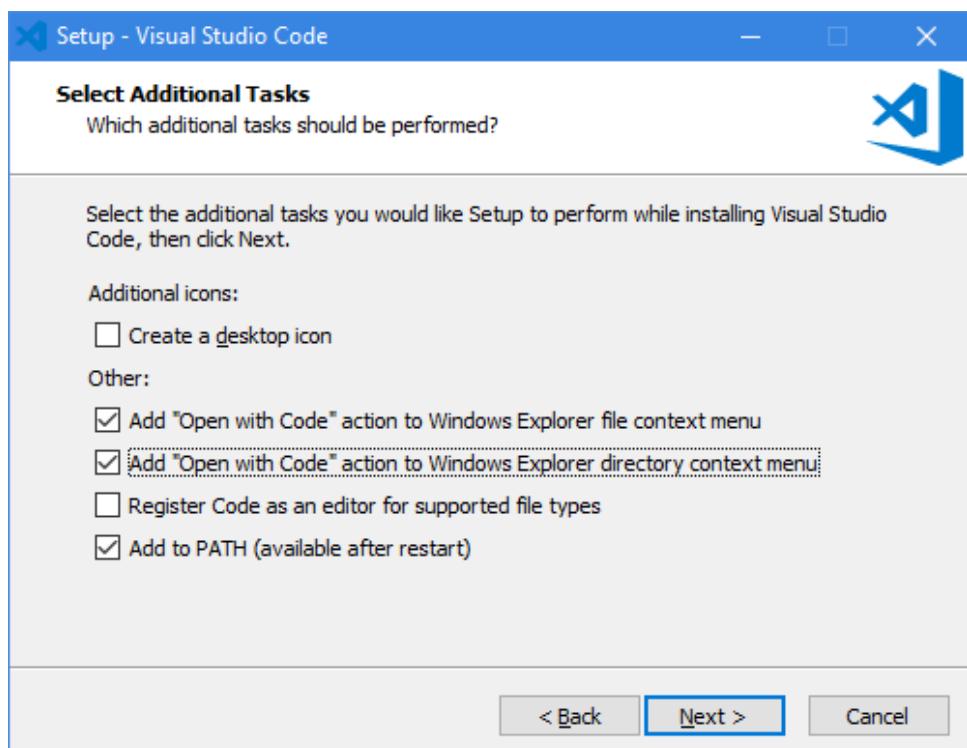
- Click on “I accept the agreement”.
- Click on “Next”.



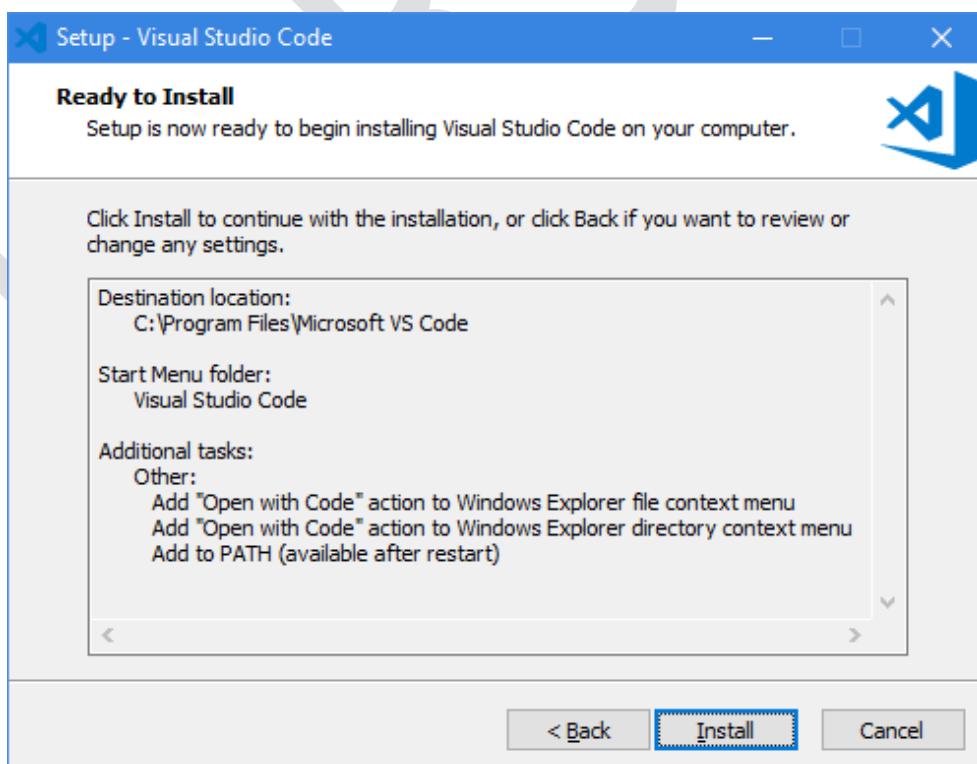
- Click on "Next".



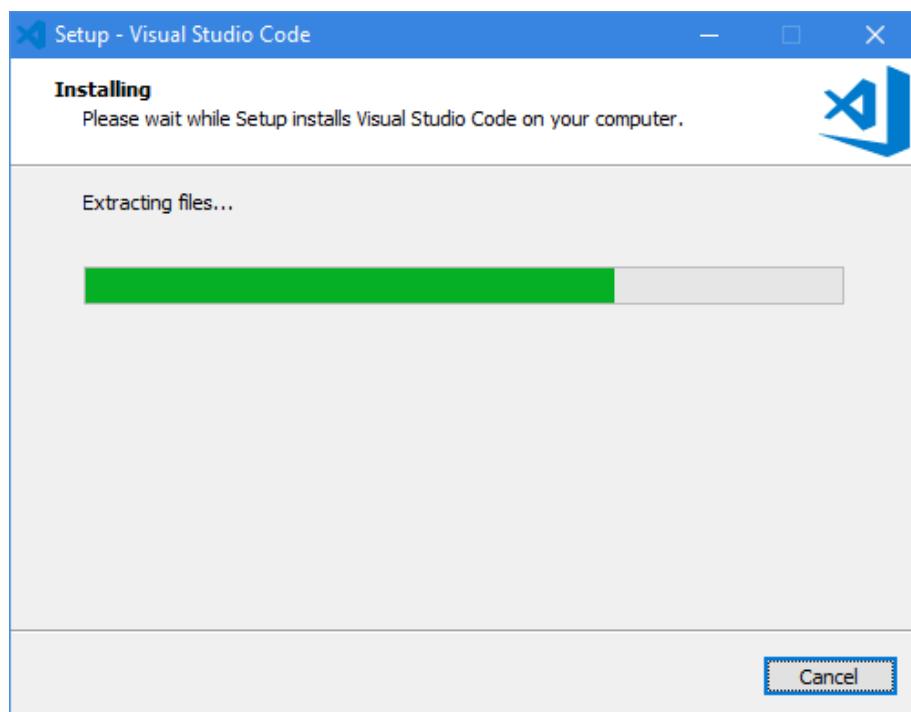
- Click on "Next".



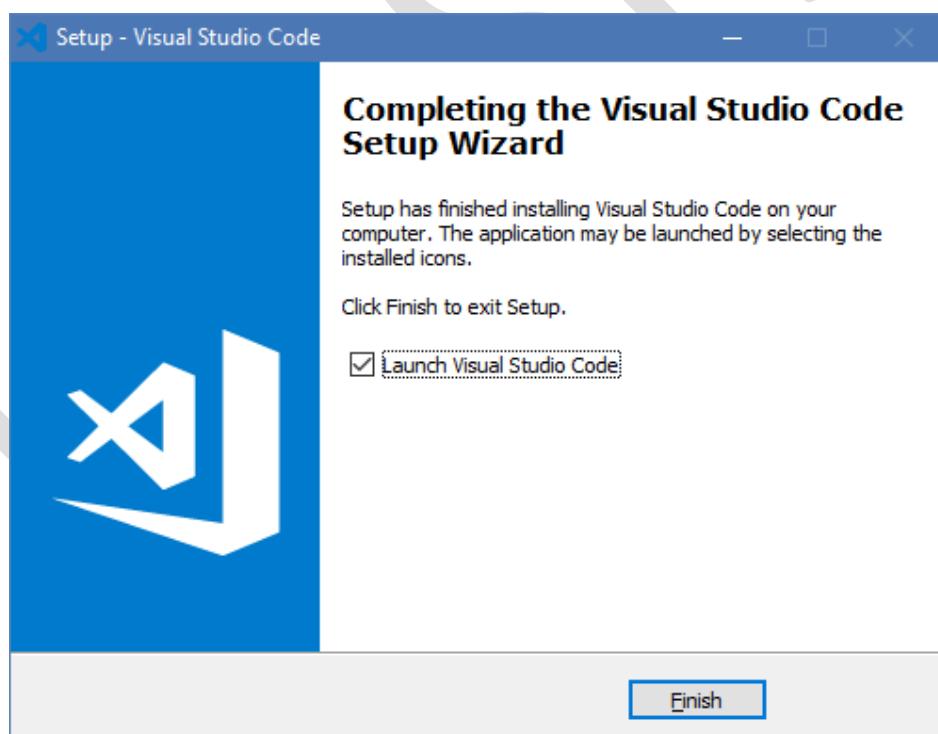
- Check the checkbox “Add Open with Code action to Windows Explorer file context menu”.
- Check the checkbox “Add Open with Code action to Windows Explorer directory context menu”.
- Click on “Next”.



- Click on “Install”.



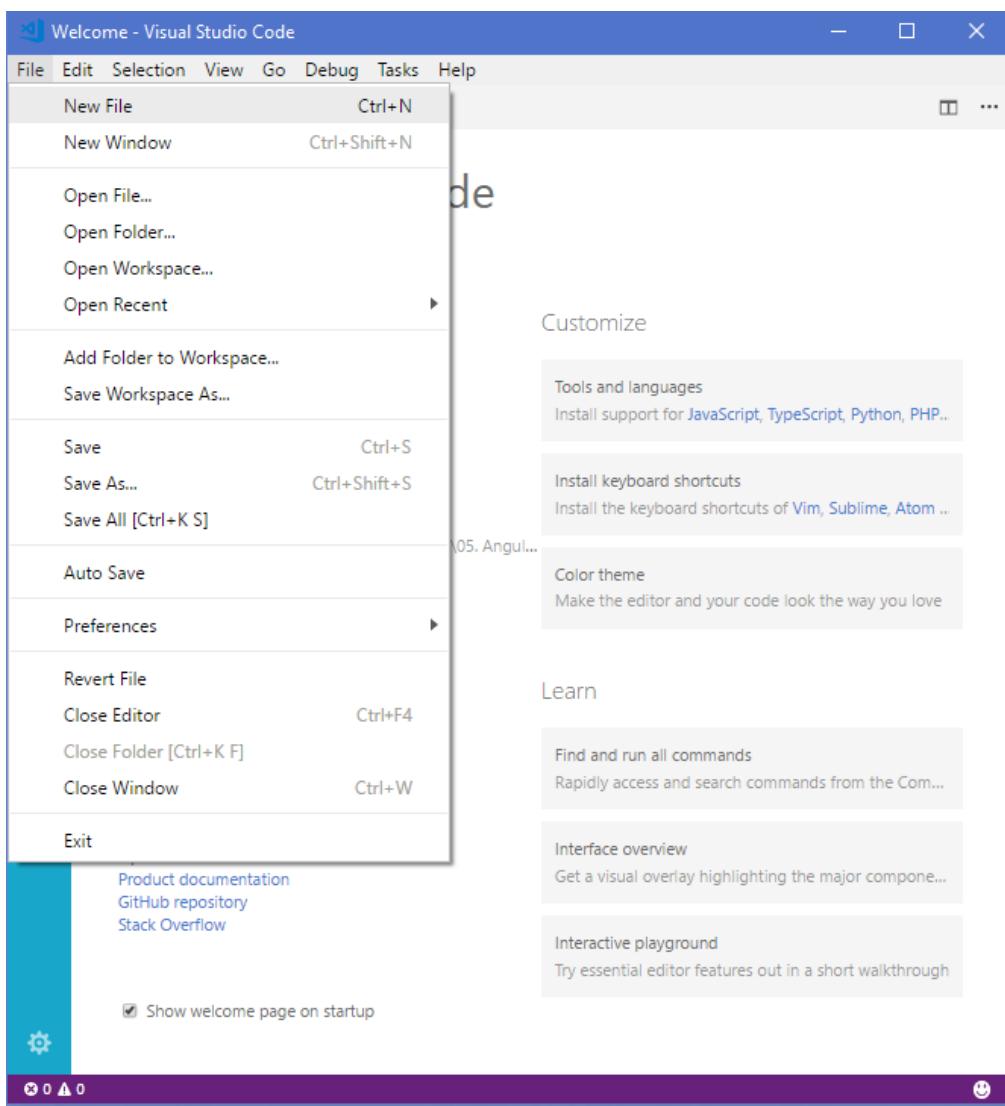
- Installation is going on....



- Click on "Finish".

5) Create TypeScript Program

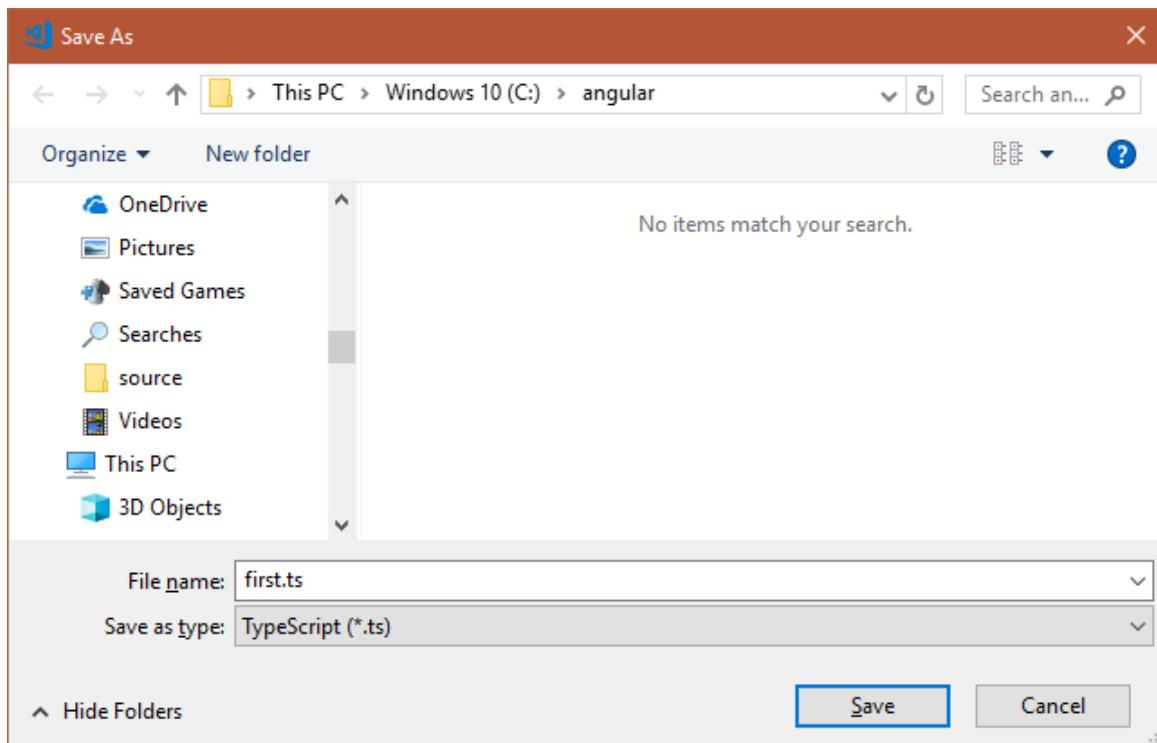
- Open "Visual Studio Code", by clicking on "Start" – "Visual Studio Code" – "Visual Studio Code".
- Visual Studio Code opened.



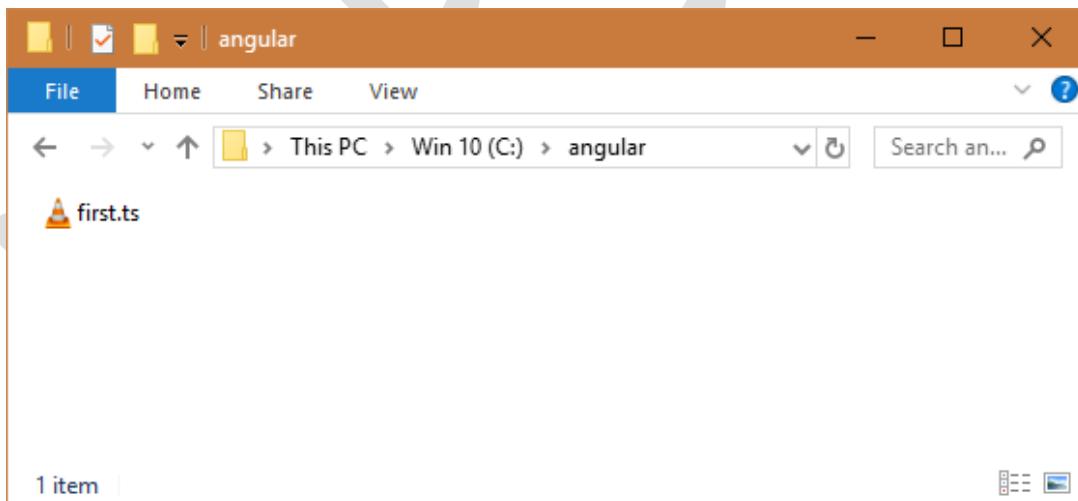
- Go to “File” – “New File”.
- Type the program as follows:

The screenshot shows a Visual Studio Code window with a TypeScript file named 'first.ts'. The code contains the following TypeScript code:
`var s : string = "Hello";
console.log(s);`

- Go to “File” menu – “Save” (or) Press Ctrl+S.



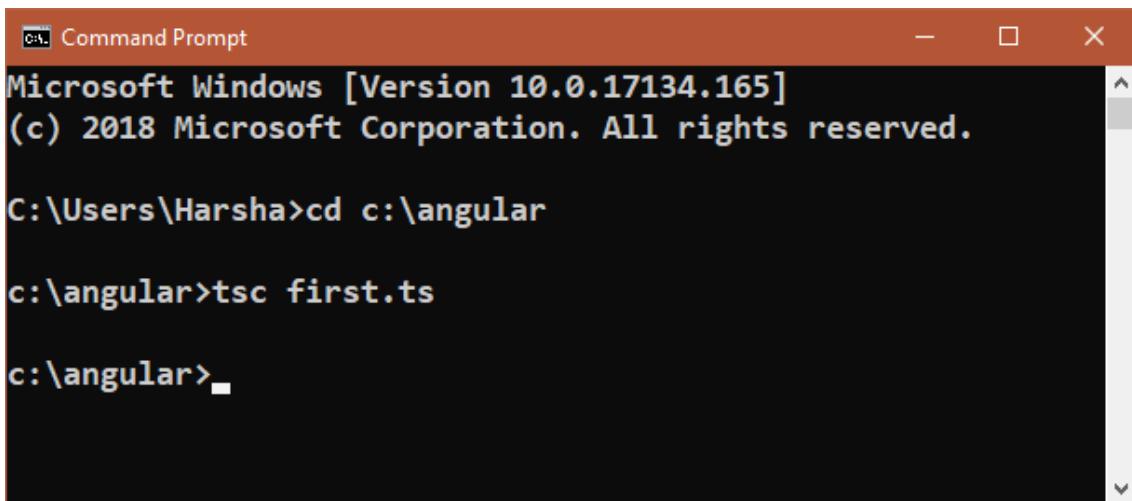
- Select "c:\angular" folder and enter the filename as "first.ts".
- Click on "Save".
- Now the typescript file (c:\angular\first.ts) is ready.



6) Compile the TypeScript Program

- Open Command Prompt and enter the following commands:

```
cd c:\angular  
tsc first.ts
```



Command Prompt
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Harsha>cd c:\angular

c:\angular>tsc first.ts

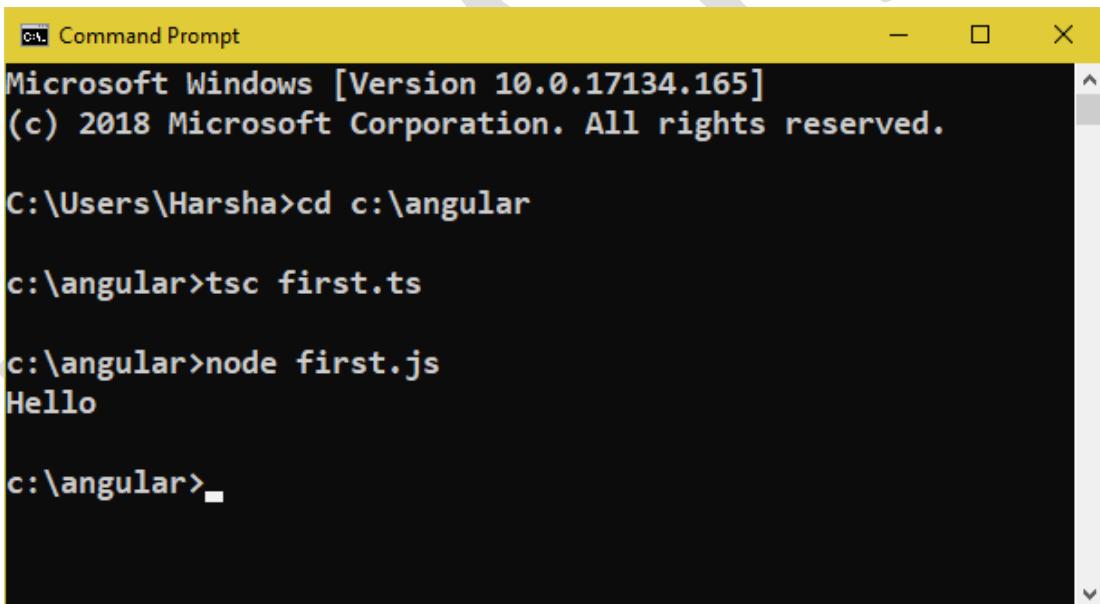
c:\angular>■

- Now the “typescript program” has been compiled into “javascript program”; So “first.js” file has been generated in “c:\angular” folder.

7) Execute the TypeScript Program

- Enter the following commands in the same Command Prompt window:

```
node first.js
```



Command Prompt
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Harsha>cd c:\angular

c:\angular>tsc first.ts

c:\angular>node first.js
Hello

c:\angular>■

Output:

Hello

- Note:** The "node" command (Provided by Node.js) executes the javascript file and executes its output in the Command Prompt itself.

TypeScript Basics

Variables

- Variable is a named memory location in RAM, to store a value at run time.
- **Syntax:** var variable : datatype = value;
- **Example:** var a : number = 100;
- TypeScript supports “optional static typing”. That means it is optional to specify datatype for the variable in TypeScript.
 - **Static Typing:** If you specify the data type for the variable, it is not possible assign other data type of value into the variable; if you do so, “tsc” compiler will generate coding-time and compile-time errors; but the code will be compiled and executed also, even though it is having errors.
 - **Dynamic Typing:** If you don’t specify the data type for the variable, we can assign any type of value into the variable.
- In TypeScript, we have “optional static typing”. That means it is optional to specify datatype for the variable in TypeScript.

Data Types

- “Data type” specifies what type of value that can be stored in a variable.
- List of TypeScript Data Types:
 1. **number:** All types of numbers (integer / floating-point numbers). Ex: 10, 10.3498
 2. **string:** Collection of characters in double quotes or single quotes. Ex: “hello”
 3. **boolean:** true or false
 4. **any:** Any type of value

Variables and Data Types - Example

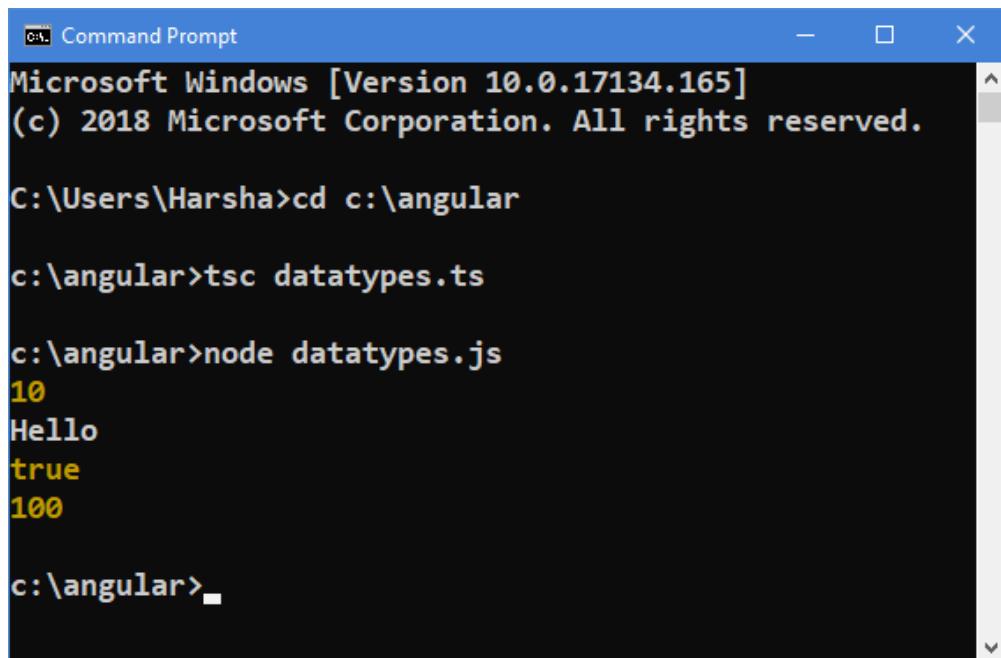
c:\angular\datatypes.ts

```
var a: number = 10;
var b: string = "Hello";
var c: boolean = true;
var d: any = 100;

console.log(a);
console.log(b);
console.log(c);
console.log(d);
```

Compilation and Execution

- Open Command Prompt
- ```
cd c:\angular
tsc datatypes.ts
node datatypes.js
```



```

Command Prompt
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Harsha>cd c:\angular

c:\angular>tsc datatypes.ts

c:\angular>node datatypes.js
10
Hello
true
100

c:\angular>_

```

## TypeScript OOP

### Object

#### What is Object

- Object is the primary concept in OOP (Object Oriented Programming).
- “Object” represents a physical item, that represents a person or a thing.
- Object is a collection of properties (details) and methods (manipulations).
- For example, a student is an "object".
- We can create any no. of objects inside the program.

#### What is Property

- Properties are the details about the object.
- Properties are used to store a value.
- For example, **studentname="Scott"** is a property in the "student object".
- Properties are stored inside the object.
- The value of property can be changed any no. of times.

#### What is Method

- Methods are the operations / tasks of the object, which manipulates / calculates the data and do some process.
- Method is a function in the object. In other words, a function which is stored inside the object is called as "Method".
- Methods are stored inside the object.

## Creating Object

- Object can be created by using "Object Literal Pattern" in TypeScript.
- "Object literal" is a collection of properties and methods, that are enclosed within curly braces {}.
- **Syntax to create Object:** { property: value, ..., method: function() { code here } }

## Reference Variable

- The "reference variable" is a variable, that can store the reference of an object.
- We can access all the properties and methods of the object, by using the "reference variable".
- **Syntax to create Object and store its reference in the "reference variable":**

```
var referenceVariable = { property: value, ..., method: function() { code here } };
```

## Objects - Example

c:\angular\objects.ts

```
var student = {
 studentId: 1,
 studentName: "Scott",
 marks: 80,
 getResult: function()
 {
 if (this.marks >= 35)
 {
 return "Pass";
 }
 else
 {
 return "Fail";
 }
 }
};

console.log(student);
console.log(student.studentId);
console.log(student.studentName);
console.log(student.marks);
console.log(student.getResult());
```

## Compilation and Execution

- Open Command Prompt
- ```
cd c:\angular  
tsc objects.ts  
node objects.js
```

```
c:\Users\Harsha>cd c:\angular  
c:\angular>tsc objects.ts  
c:\angular>node objects.js  
{ studentId: 1,  
  studentName: 'Scott',  
  marks: 80,  
  getResult: [Function: getResult] }  
1  
Scott  
80  
Pass  
c:\angular>
```

Class

What is Class

- “Class” represents a model of the object, which defines list of properties and methods of the object.
- **Ex:** “Student” class represents structure (list of properties and methods) of every student object.
- We can create any no. of objects based on a class, using “new” keyword.
- All the objects of a class, shares same set of properties & methods of the class.
- We can store the reference of the object in “reference variable”; using which you can access the object.

```
class classname  
{  
  properties  
  methods  
}
```

Steps for working with classes and objects

- Create a Class:

```
class classname
{
    property: datatype = value;
    ...
    methodname( parameter1: datatype, parameter2: datatype, ... ) : returntype
    {
        code here
    }
    ...
}
```

- Create an Object & Store its address in Reference Variable

```
var referencevariablename = new classname();
```

- Access Properties and Methods using Reference Variable

```
referencevariablename.property  
referencevariablename.method();
```

Classes - Example

```
c:\angular\classes.ts
class Student
{
    studentId: number;
    studentName: string;
    marks: number;

    getResult()
    {
        if (this.marks >= 35)
        {
            return "Pass";
        }
        else
        {
            return "Fail";
        }
    }
}

var s1 = new Student();
```

```
s1.studentId = 101;
s1.studentName = "Scott";
s1.marks = 80;
console.log(s1.studentId);
console.log(s1.studentName);
console.log(s1.marks);
console.log(s1.getResult());

var s2 = new Student();
s2.studentId = 102;
s2.studentName = "Smith";
s2.marks = 72;
console.log(s2.studentId);
console.log(s2.studentName);
console.log(s2.marks);
console.log(s2.getResult());
```

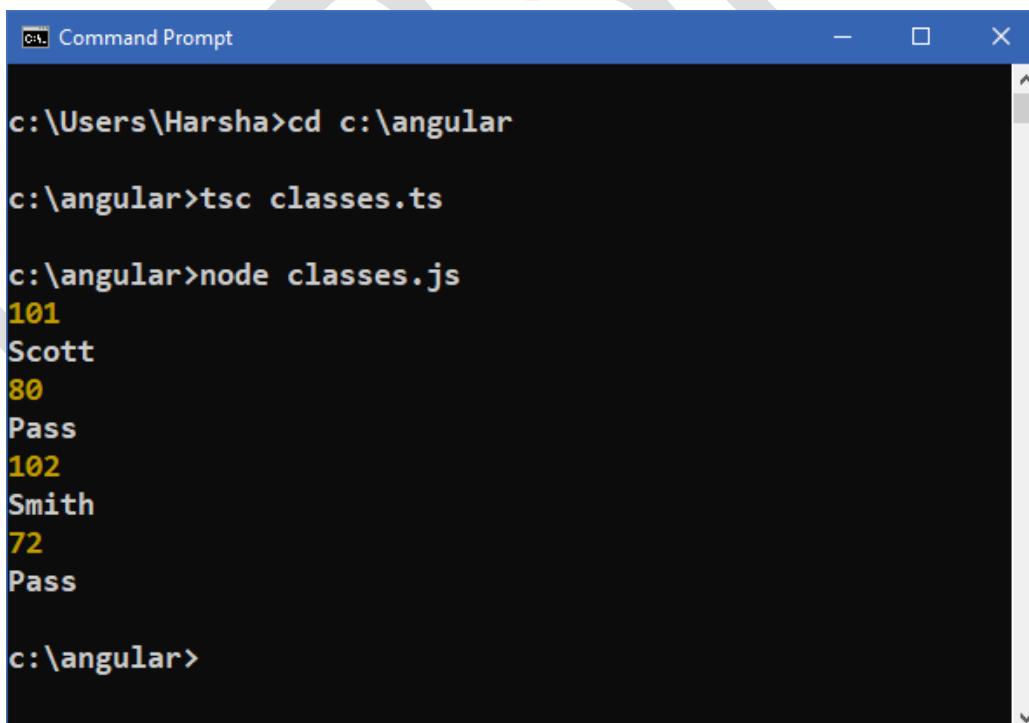
Compilation and Execution

- Open Command Prompt

```
cd c:\angular
```

```
tsc classes.ts
```

```
node classes.js
```



The screenshot shows a Windows Command Prompt window titled "Command Prompt". The command history and output are as follows:

```
c:\Users\Harsha>cd c:\angular
c:\angular>tsc classes.ts
c:\angular>node classes.js
101
Scott
80
Pass
102
Smith
72
Pass

c:\angular>
```

Constructor

- Constructor is a special function which is a part of the class.
- Constructor will be called automatically when we create a new object for the class. If you create multiple objects for the same class, the constructor will be called each time when you create new object.

- Constructor is used to initialize properties of the class.
- Constructor's name should be "constructor".
- Constructor can receive arguments; but can't return any value.
- In TypeScript, we can't define multiple constructors.

Syntax of Constructor

```
constructor(parameter1: dataType, ...)  
{  
    code here  
}
```

Constructor - Example

```
c:\angular\constructor.ts  
class Student  
{  
    studentId: number;  
    studentName: string;  
    marks: number;  
  
    constructor(studentId: number = 0, studentName: string = "none", marks: number = 0)  
    {  
        this.studentId = studentId;  
        this.studentName = studentName;  
        this.marks = marks;  
    }  
  
    getResult()  
    {  
        if (this.marks >= 35)  
        {  
            return "Pass";  
        }  
        else  
        {  
            return "Fail";  
        }  
    }  
}  
  
var s1 = new Student(101, "Scott", 80);  
console.log(s1.studentId);  
console.log(s1.studentName);  
console.log(s1.marks);  
console.log(s1.getResult());  
  
var s2 = new Student(102, "Smith", 45);  
console.log(s2.studentId);  
console.log(s2.studentName);
```

```

console.log(s2.marks);
console.log(s2.getResult());

var s3 = new Student();
console.log(s3.studentId);
console.log(s3.studentName);
console.log(s3.marks);
console.log(s3.getResult());

```

Compilation and Execution

- Open Command Prompt

```

cd c:\angular
tsc constructor.ts
node constructor.js

```



A screenshot of a Microsoft Windows Command Prompt window titled "Command Prompt". The window shows the following output:

```

Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Harsha>cd c:\angular

c:\angular>tsc constructor.ts

c:\angular>node constructor.js
101
Scott
80
Pass
102
Smith
45
Pass
0
none
0
Fail

c:\angular>_

```

Inheritance

- Inheritance is a concept of extending the parent class, by creating the child class. The child class extends parent class. That means all the members of parent class will be inherited (derived) into the child class.

- When you create an object for the child class, it includes all the members (properties and methods) of both child class and parent class. When you create an object for parent class, it includes all the members (properties and methods) of parent class only.
- For example, the “Student” class extends “Person class”. The “Car” class extends “Vehicle” class.
- When parent class has a constructor, the child class’s constructor must call the parent class’s constructor explicitly.
- The “extends” keyword is used to create inheritance.
- The “super” keyword represents current parent class. It can be used to call the constructor or method of parent class.

Types of Inheritance

- **Single Inheritance:** One parent class with one child class.
- **Multiple Inheritance:** Multiple parent classes with one child class.
- **Hierarchical Inheritance:** One parent class with multiple child classes.
- **Multi-Level Inheritance:** One parent class with one child class and the child class has another child class.
- **Hybrid Inheritance:** One parent class with one child class.

Syntax of Inheritance

- **Create parent class:**

```
class parentclass  
{  
    parent class's members  
}
```

- **Create child class:**

```
class childclass extends parentclass  
{  
    child class's members  
}
```

- **Create object for parent class:**

```
var referencevariable = new parentclass();  
referencevariable.parentclassmember
```

- **Create object for child class:**

```
var referencevariable = new childclass();  
referencevariable.childclassmember  
referencevariable.parentclassmember
```

Inheritance - Example

c:\angular\inheritance.ts

```
class Person
{
  name: string;
  age: number;

  constructor(a: string, b: number)
  {
    this.name = a;
    this.age = b;
  }

  getDetails(): string
  {
    return this.name + ", " + this.age;
  }
}

class Student extends Person
{
  studentid: number;
  marks: number;

  constructor(p: string, q: number, r: number, s: number)
  {
    super(p, q);
    this.studentid = r;
    this.marks = s;
  }

  getDetails(): string
  {
    return this.name + ", " + this.age + ", " + this.studentid + ", " + this.marks;
  }
}

var p = new Person("Scott", 20);
console.log(p.name);
console.log(p.age);
console.log(p.getDetails());

console.log("-----");
var s = new Student("Smith", 22, 201, 67);
console.log(s.name);
console.log(s.age);
console.log(s.studentid);
console.log(s.marks);
console.log(s.getDetails());
```

Compilation and Execution

- Open Command Prompt

```
cd c:\angular  
tsc inheritance.ts  
node inheritance.js
```

The screenshot shows a Windows Command Prompt window titled "Command Prompt". The command history and output are as follows:

```
c:\Users\Harsha>cd c:\angular  
c:\angular>tsc inheritance.ts  
c:\angular>node inheritance.js  
Scott  
20  
Scott, 20  
-----  
Smith  
22  
201  
67  
Smith, 22, 201, 67  
c:\angular>
```

The output shows the execution of the TypeScript file, which defines a class named Person with properties name and age, and a constructor that initializes both. It then creates two instances of Person: one for "Scott" (age 20) and one for "Smith" (age 201). Both instances have their properties printed to the console, along with a separator line and the final output.

Access Modifiers

- Access Modifiers specify where the member of a class can be accessible.
- That means it specifies whether the member of a class is accessible outside the class or not.
- These are used to implement "security" in OOP.
- For each member (property / method), we can specify the access modifier separately.
- TypeScript compiler and Visual Studio Code Editor shows errors, if a developer try to access the member, which is not accessible.
- "public" is the access modifier for all the members (property / method) in TypeScript class.
- TypeScript supports three access modifiers:
 1. public (default)
 2. private
 3. protected

1. public (default):

- The public members are accessible anywhere in the program (in the same class and outside the class also).

2. private:

- The private members are accessible within the same class only; If you try to access them outside the class, you will get compile-time error.

3. protected:

- The protected members are accessible within the same class and also in the corresponding child classes; If you try to access them outside the same class or child class, you will get compile-time error.

Syntax of creating a member with access modifier:**• Property:**

```
class classname
{
    accessmodifier property: datatype;
}
```

• Method:

```
class classname
{
    accessmodifier methodname( arguments) : returntype
    {
    }
}
```

Access Modifiers - Example

c:\angular\accessmodifiers.ts

```
/* Current class */
class Student
{
    public studentid: number = 101;
    private studentname: string = "Scott";
    protected marks: number = 80;

    public display(): void
    {
        console.log("Student.display:");
        console.log(this.studentid); //accessible
        console.log(this.studentname); //accessible
        console.log(this.marks); //accessible
    }
}

/* Child class */
class EngineeringStudent extends Student
```

```
{  
  public display2():void  
  {  
    console.log("EngineeringStudent.display2:");  
    console.log(this.studentid); //accessible  
    //console.log(this.studentname); //not accessible  
    console.log(this.marks); //accessible  
  }  
}  
  
/* Other class */  
class Test  
{  
  sampleMethod()  
  {  
    var s = new Student();  
    s.display(); //accessible  
    var s2 = new EngineeringStudent();  
    s2.display2(); //accessible  
    console.log("-----");  
    console.log(s.studentid); //accessible  
    //console.log(s.studentname); //not accessible  
    //console.log(s.marks); //not accessible  
  }  
}  
  
var t:Test = new Test();  
t.sampleMethod();
```

Compilation and Execution

- Open Command Prompt

```
cd c:\angular  
tsc accessmodifiers.ts  
node accessmodifiers.js
```

```
c:\Users\Harsha>cd c:\angular  
c:\angular>tsc accessmodifiers.ts  
c:\angular>node accessmodifiers.js  
Student.display:  
101  
Scott  
80  
EngineeringStudent.display2:  
101  
80  
-----  
101  
c:\angular>■
```

Interfaces

- Interface is the model of a class, which describes the list of properties and methods of a class.
- Interfaces doesn't contain actual code; contains only list of properties and methods.
- Interfaces doesn't contain method implementation (method definition); it contains method declaration only, which defines method access modifier, method name, method arguments and method return type.
- The child class that implements the interface must implement all the methods of the interface; if not, compile-time error will be shown at child class. The method name, parameters, return type and access modifier must be same in between "interface method (method at the interface)" and "class method (method at the class)".
- Interfaces act as a mediator between two or more developers; one developer implements the interface, other developer creates reference variable for the interface and invokes methods; so interface is common among them.
- The child class can implement the interface with "implements" keyword.
- All the methods of interface is by default, "public".
- One interface can be implemented by multiple classes; One class can implement multiple interfaces.
- We can develop "multiple inheritance" by implementing multiple interfaces at-a-time in the same class.

Steps for development of Interfaces

- Create an interface:

```
interface interfacename  
{  
    property: datatype;
```

```
...
    method( arguments ): returntype;
...
}
```

- **Create a child class for the interface and implement it:**

```
class classname implements interfacename
{
    property: datatype;

    method( arguments ) : returntype
    {
        code here
    }
}
```

Interfaces - Example

```
c:\angular\interfaces.ts
interface IStudent
{
    firstName: string;
    lastName: string;
    getFullName(): string;
}

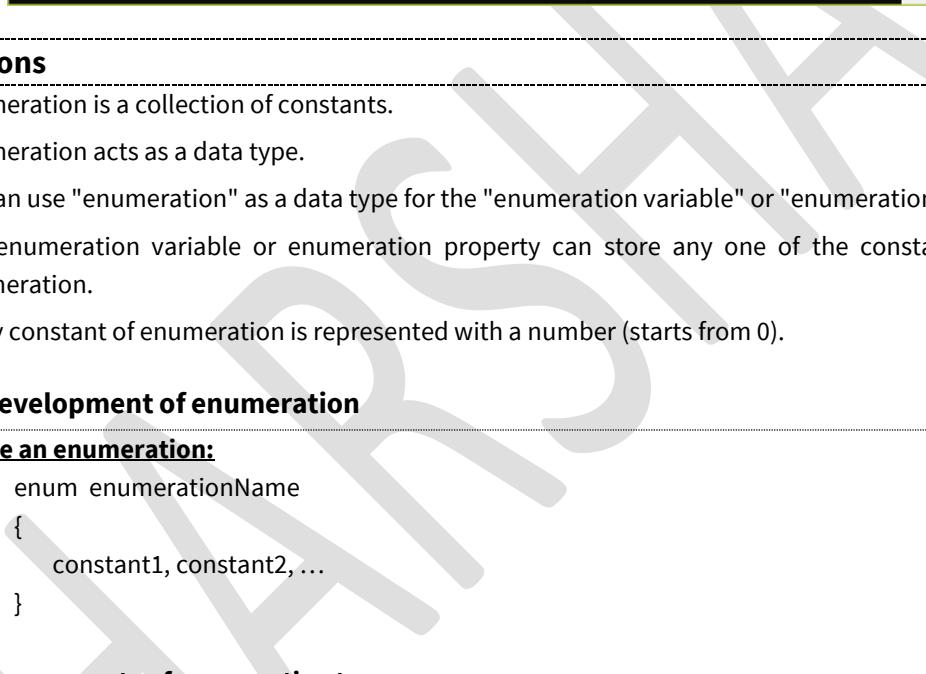
class Student implements IStudent
{
    firstName: string;
    lastName: string;

    getFullName(): string
    {
        return this.firstName + " " + this.lastName;
    }
}
var s = new Student();
s.firstName = "Adam";
s.lastName = "Smith";
console.log(s.getFullName());
```

Compilation and Execution

- Open Command Prompt

```
cd c:\angular
tsc interfaces.ts
node interfaces.js
```



```
c:\ Command Prompt
c:\Users\Harsha>cd c:\angular
c:\angular>tsc interfaces.ts
c:\angular>node interfaces
Adam Smith
c:\angular>
```

Enumerations

- Enumeration is a collection of constants.
- Enumeration acts as a data type.
- We can use "enumeration" as a data type for the "enumeration variable" or "enumeration property".
- The enumeration variable or enumeration property can store any one of the constants of the same enumeration.
- Every constant of enumeration is represented with a number (starts from 0).

Steps for development of enumeration

- **Create an enumeration:**

```
enum enumerationName
{
    constant1, constant2, ...
}
```

- **Create a property of enumeration type:**

```
class classname
{
    property: enumerationName;
}
```

- **Create a variable of enumeration type:**

```
variableName: enumerationName;
```

- **Assign the value into enumeration variable or enumeration property:**

```
enumerationVariable = enumerationName.constant;
this.enumerationProperty = enumerationName.constant;
```

Enumerations - Example

c:\angular\enumerations.ts

```
enum courseNames
{
    Java, Blockchain, JavaScript
}

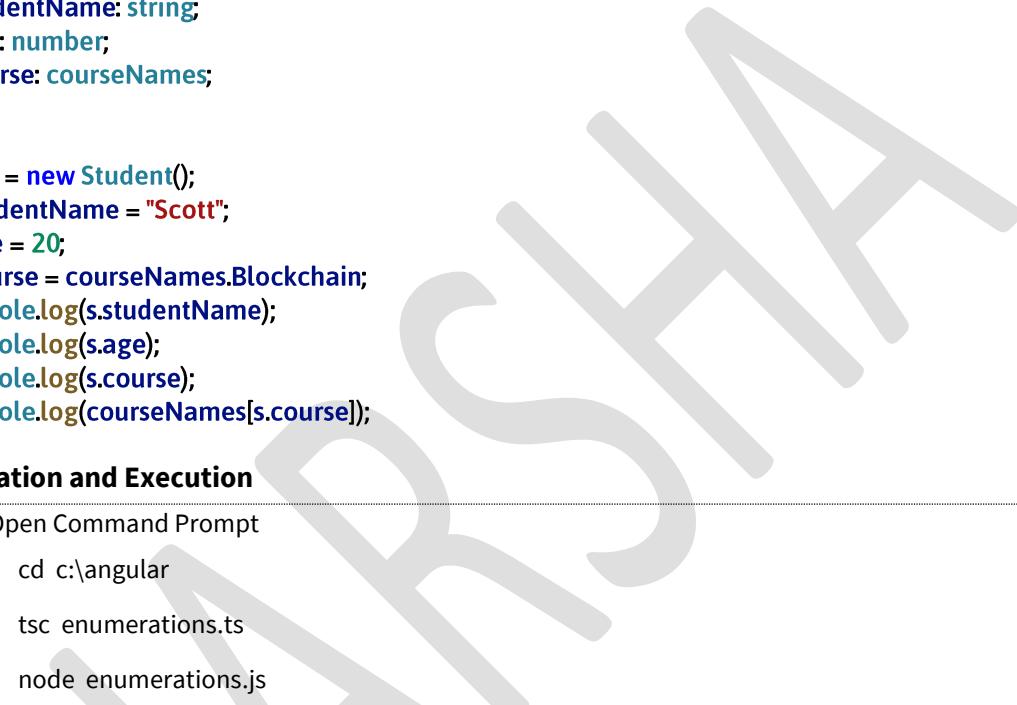
class Student
{
    studentName: string;
    age: number;
    course: courseNames;
}

var s = new Student();
s.studentName = "Scott";
s.age = 20;
s.course = courseNames.Blockchain;
console.log(s.studentName);
console.log(s.age);
console.log(s.course);
console.log(courseNames[s.course]);
```

Compilation and Execution

- Open Command Prompt

```
cd c:\angular
tsc enumerations.ts
node enumerations.js
```



```
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Harsha>cd c:\angular

c:\angular>tsc enumerations.ts

c:\angular>node enumerations.js
Scott
20
1
Blockchain

c:\angular>
```

Modules

- In large scale applications, it is recommended to write each class in a separate file.
- To access the class of one file in other file, we need the concept of "Modules".
- Module is a file (typescript file), which can export one or more classes (or interfaces, enumerations etc.) to other files; The other files can import classes (or interfaces, enumerations etc.), that are exported by the specific file. We can't import the classes (or others) that are not exported.
- We can export the class (or others), by using "export" keyword in the source file.
- We can import the class (or others), by using "import" keyword in the destination file.
- To represent:
 1. current folder, we use "./".
 2. sub folder in current folder, we use "./subfolder".
 3. parent folder, we use "../".

Steps for development of modules

- Export a class in file1.ts

```
export class className1
{
  ...
}
```

- Import class in file2.ts:

```
import { className1 } from "./file1.ts";
class className2
{
  ...
}
```

Modules - Example

c:\angular\Student.ts

```
export class Student
{
  studentId: number;
  studentName: string;
  marks: number;

  constructor(studentId: number, studentName: string, marks: number)
  {
    this.studentId = studentId;
    this.studentName = studentName;
    this.marks = marks;
  }
}
```

c:\angular\StudentsList.ts

```
import { Student } from "./Student";
```

```
class StudentsList
{
  students: Student[] = [
    new Student(101, "Scott", 80),
    new Student(102, "Smith", 45),
    new Student(103, "Allen", 97),
    new Student(104, "Jones", 25)
  ];

  getTopRanker(): Student
  {
    var index;
    var max = 0;
    for (var i in this.students)
    {
      if (this.students[i].marks > max)
      {
        max = this.students[i].marks;
        index = i;
      }
    }
    return this.students[index];
  }

  var s: StudentsList = new StudentsList();
  console.log(s.getTopRanker());
}
```

Compilation and Execution

- Open Command Prompt

```
cd c:\angular
tsc Student.ts
tsc StudentsList.ts
node StudentsList.js
```



```
c:\ Command Prompt
c:\Users\Harsha>cd c:\angular
c:\angular>tsc Student.ts
c:\angular>tsc StudentsList.ts
c:\angular>node StudentsList.js
Student { studentId: 103, studentName: 'Allen', marks: 97 }
c:\angular>
```

ANGULAR 7

Introduction to Angular 7

What is Angular?

- Angular is a client side framework, which is used to create web applications. The framework provides skeleton of the project and specifies clear guidelines, where to write which type of code.
- Angular can be used in combination with any server side platform such as Java, NodeJS, Asp.Net, PHP, Python etc.
- Angular is developed using "TypeScript" language, which is a superset of JavaScript language.
- Angular is the most-used client side framework.
- Angular was developed by Google.
- Angular is free-to-use (commercially too).
- Angular is open-source. That means the source code of angular is available online for free of cost.
- Angular is cross-platform. That means it works in all the operating systems.
- Angular is cross-browser compatible. That means it works in all the browsers, except less than IE 9 (which is completely out-dated).
- Angular is mainly used to create "data bindings". That means, we establish relation between a variable and html element; When the value of the variable is changed, the same will be automatically effected in the corresponding html element; and vice versa. So that the developer need not write any code for DOM manipulations (updating values of html tags, based on user requirements. for example, updating the list of categories when a new category added by the user). Thus the developer can fully concentrate on the application logic, instead of writing huge code for DOM manipulations. So we can achieve clean separation between "application logic" and "DOM manipulations".
- Angular mainly works based on "Components". The component is a class, which represents a specific section (part) of the web page.

Goals of Angular

- Separation of DOM manipulation from application logic.
- Separation of HTML logic from application logic.
- Make SPA (Single Page Application) development easier. SPA provides client-side navigation system; but can communicate with server only through AJAX; the web page never gets refreshed fully. Ex: Gmail.
- Separation of business logic from application logic.
- Enable Unit testing. The components can be unit tested individually.

Versions of Angular

- Angular 2 : Sep 14th 2016
- Angular 4 : Mar 23rd 2017 [Angular 3.0 was skipped due to misalignment of router package 3.3.0]
- Angular 5 : Nov 1st 2017
- Angular 6 : May 4th 2018

Angular (vs) AngularJS

- “Angular 2” is not an extension to “Angular 1”; it is a completely “rebuilt” or “rewrite”. “Angular 2” is rewrite from the same team that built “AngularJS 1”.
- “Angular 4”, “Angular 5” and “Angular 6” are mostly similar to “Angular 2”, except some internal changes for stability and performance.
- “Angular 2+” means “Angular 2, Angular 4 and Angular 5”. **Note:** “Angular 3” is not released, to avoid the confusion of “router” package, which is already released with “3.3.0” version.

| Sl. No | AngularJS | Angular |
|-------------------|---|---|
| 1 | “AngularJS” has performance drawbacks because, the “digest loop” checks and updates all properties when a field or property has been modified. The digest loop may repeat up to 10 times; due to a watcher of one property can update other property. | “Angular” is faster in performance because it updates a single property that has been really modified. No digest loop and no need of any repetition. |
| 2 | “AngularJS” is based on “JavaScript”, which is “prototype-based OOP language”. | “Angular” is based on “TypeScript”, which is “class-based OOP language”. |
| 3 | “AngularJS” is based on “MVC architecture”. The code will be divided into three major parts called “Model”, “View” and “Controller”. Model is an object that stores data. View is a html file that contains presentation logic to display data. Controller is a function that manipulates data. | “Angular” is based on “Components”. Component is a class that contains stores data and also manipulates the data. View is a html file that contains presentation logic to display data. |
| 4 | “AngularJS” is mainly used for development of “Single Page Applications”. | “Angular” is also mainly used for development of “Single Page Applications”. |
| 5 | Apart from data bindings, Angular 1 provides many additional features such as validations, routing, animations, services, filters, providers, factories, configuration, AJAX etc. | Apart from data bindings, Angular 2 provides many additional features such as validations, routing, animations, services, pipes, AJAX etc. |
| 6 | Supports “Scopes” (Models) to store data. | Doesn’t supports Scopes; but supports Components alternatively. |
| 7 | Supports only one expression syntax {{ }}. | Supports multiple expression syntaxes such as {{ }}, [], (). |

| | | |
|---|--|---|
| 8 | Doesn't provide CLI (Command Line Interface), to generate components and services easily from the command prompt commands. | Provides CLI (Command Line Interface), to generate components and services easily from the command prompt commands. |
|---|--|---|

Angular (vs) jQuery

- jQuery is a “library”, which provides a set of pre-defined functions to perform DOM manipulations directly.
- Angular is a framework, which provides the complete application structure and set of concepts to organize the client side code completely, which enables the developers to work with “Data bindings” to avoid DOM manipulations directly.

Angular (vs) React

- React just acts as a "view layer", which provides necessary concepts to create data bindings in the view.
- Angular is a "complete client side framework", which provides the complete application structure and set of concepts to organize the client side code completely, which enables the developers to work with “Data bindings” to avoid DOM manipulations directly.
- Angular provides so many concepts such as services, directives, dependency injection, routing etc., which are not supported by React.

Features of Angular

- **TypeScript based:** Pre-defined code and User-defined code is developed based on TypeScript language.
- **Faster performance:** “Angular 2+” executes faster than “AngularJS 1”.
- **Modular:** Angular is divided into multiple small parts called “packages”. For example, “core”, “common”, “forms”, “router” etc. The developer can use any of those modules based on the requirement.

Browser Compatibility of Angular 2+

- Angular 2+ supports the following browsers:

| Browser | Supported Version |
|----------------------|-------------------|
| Google Chrome | Any Version |
| Mozilla Firefox | Any Version |
| MS Internet Explorer | 9+ |
| MS Edge | 13+ |
| Safari | 7+ |
| Android Browser | 4.1+ |

Fundamentals of Angular

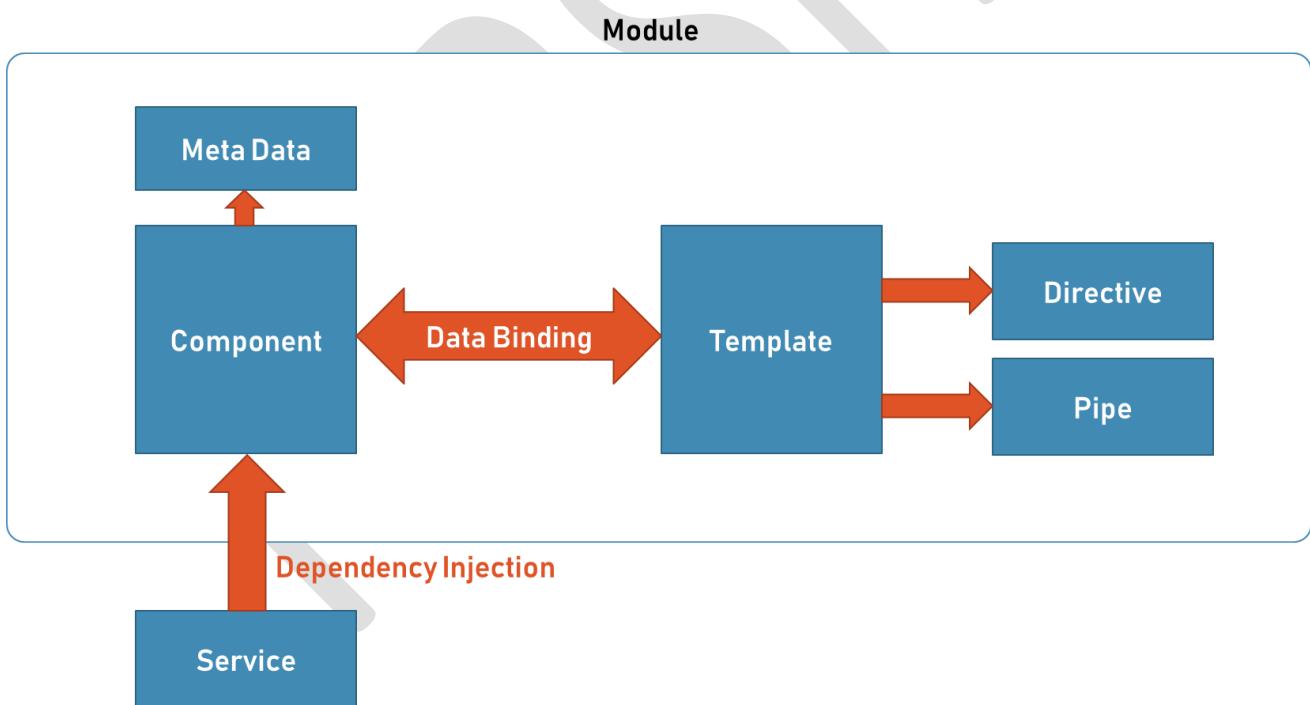
Building Blocks of Angular

- Angular is composed with the following “building blocks”:

| | | |
|--------------------------------|---|--|
| 1. Component | : | Application state + Application logic |
| 2. Metadata | : | Details about the component / module etc. |
| 3. Template | : | Design logic (HTML) |
| 4. Data Binding | : | Connection between HTML element and Component property |
| 5. Module | : | Group of components, directives and pipes. |
| 6. Service | : | Re-usable code / business logic. |
| 7. Dependency Injection | : | Injecting (loading) Service objects into Components. |
| 8. Directive | : | Manipulating DOM elements |
| 9. Pipe | : | Transforming values before displaying |

Angular Architecture

- Angular 2+ framework is built based on this architecture.



Types of compilation in Angular

- Angular framework supports two types of compilation.

Just-In-Time Compilation

- Templates (.html files) and angular compiler files will be loaded into the browser and then the templates will be compiled automatically at the time of execution, when the component is invoked.

- The template will be compiled only for the first time, when it invoked, after loading the application files into the browser.
- **Disadvantage:** Performance is slower, because every time when you run the application, the templates will be loaded into browser and compiled in the browser; it takes some time to compile.
- **Advantage:** The developer need not compile it manually at command prompt, for each modification of code.
- This is recommended during the development.
- Bootstrapping (loading app module into the browser) is done by “@angular/platform-browser-dynamic” package.

Ahead-Of-Time Compilation

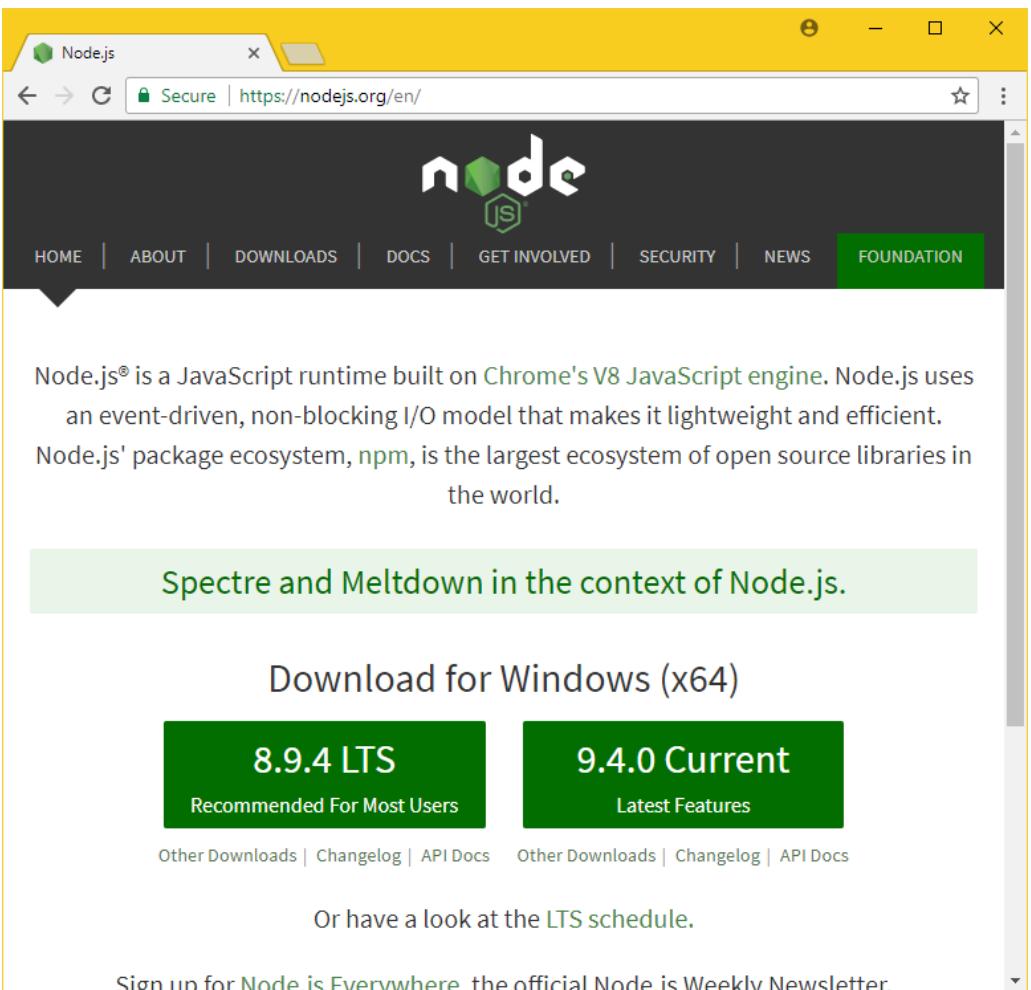
- The developer gives “ngc” command in the Command Prompt; Then the “ngc” compiler compiles the templates into javascript code; the compiled javascript code will be loaded into the browser and it directly executes. There is no need of loading “templates (.html files)” and “angular compiler scripts” into the browser.
- **Advantage:** Performance is faster, because the templates are already compiled.
- **Disadvantage:** The developer need to compile it manually at command prompt, for each modification of code.
- This is recommended in the production server only.
- Bootstrapping is done by “@angular/platform-browser” package.

Steps to Prepare First Application in Angular 2+

- 1) Installing NodeJS
- 2) Installing TypeScript
- 3) Installing Visual Studio Code
- 4) Creating Application Folder
- 5) Install “@angular/cli” package
- 6) Creating New Angular App
- 7) Open the App in Visual Studio Code
- 8) Edit code in app.component.html
- 9) Compile the application
- 10) Run the application

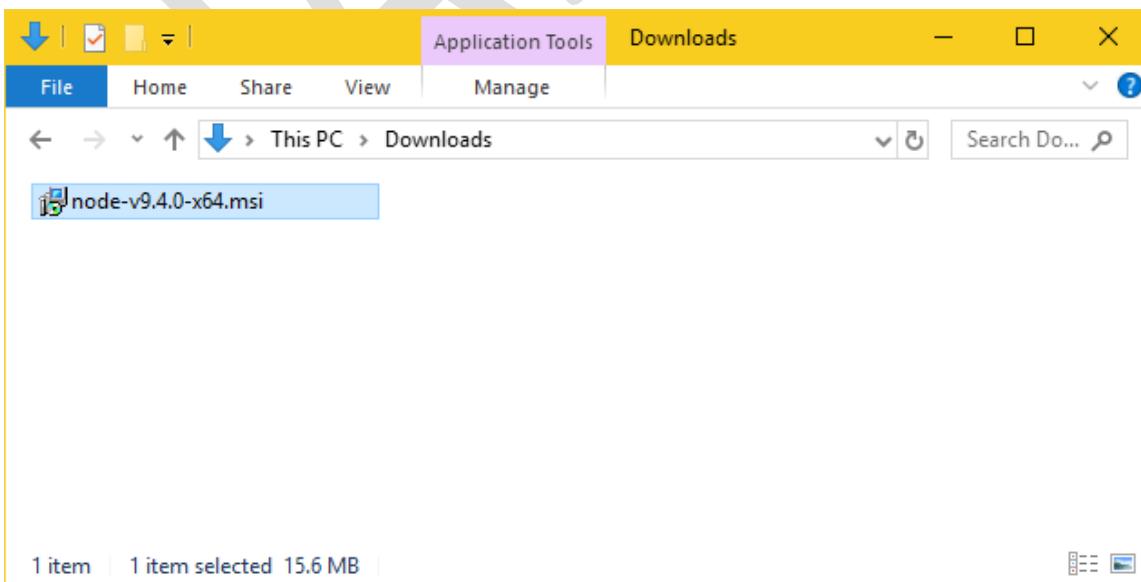
1) Installing NodeJS

- Angular 2+ framework is available as a collection of packages; those packages are available in “npm” (NodeJS Package Manager). To use “npm”, NodeJS must be installed.
- If you have already installed nodejs in your system, you can skip this step and go to step 2.
- If you have not installed nodejs in your system, you continue this step.
- Go to “<https://nodejs.org>”.

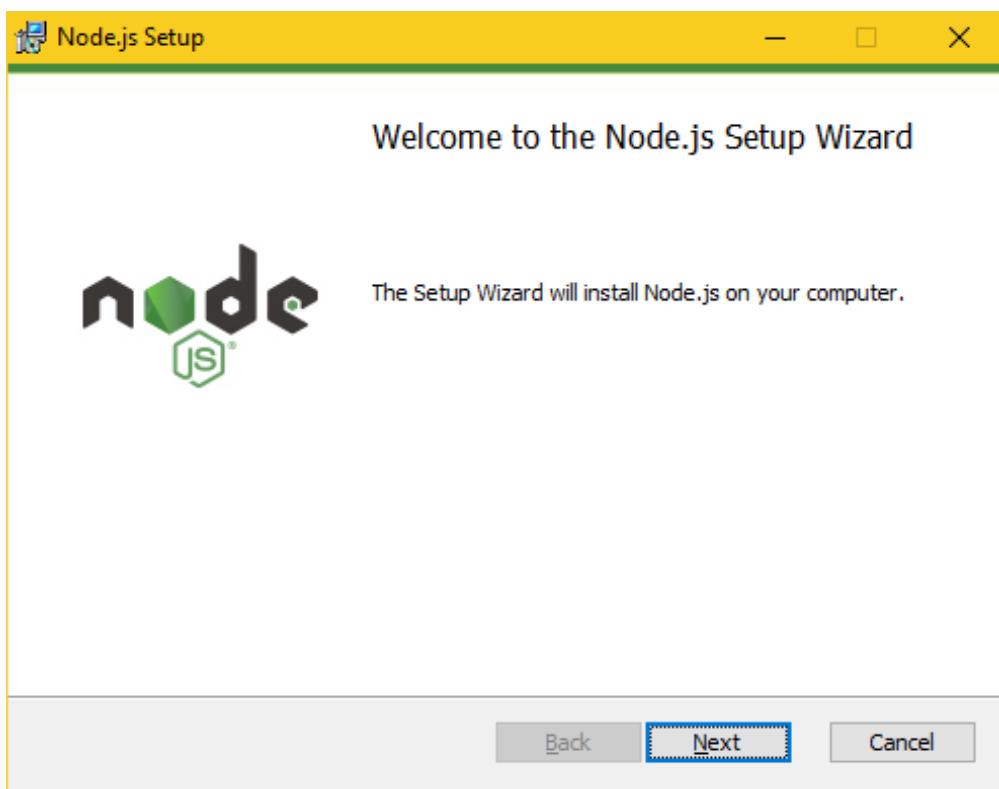


The screenshot shows the official Node.js website. At the top, there's a navigation bar with links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, NEWS, and FOUNDATION. The main content area features a brief introduction about Node.js, mentioning its event-driven, non-blocking I/O model and its large package ecosystem, npm. A green callout box highlights "Spectre and Meltdown in the context of Node.js." Below this, there are two prominent download buttons: one for "8.9.4 LTS" (Recommended For Most Users) and another for "9.4.0 Current" (Latest Features). Both buttons have links to "Other Downloads", "Changelog", and "API Docs". A note below the buttons suggests looking at the LTS schedule.

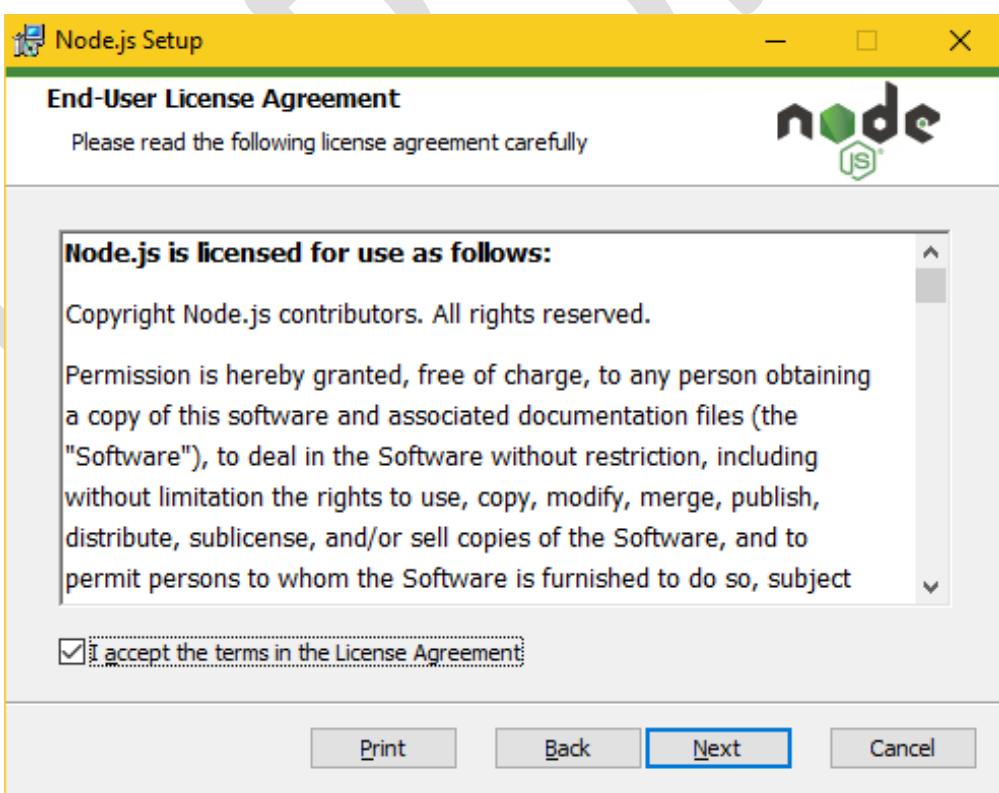
- Click on “9.4.0 Current”. **Note:** The version number can be very. Choose the latest version.
- You will download a file called “node-v9.4.0-x64.msi”.



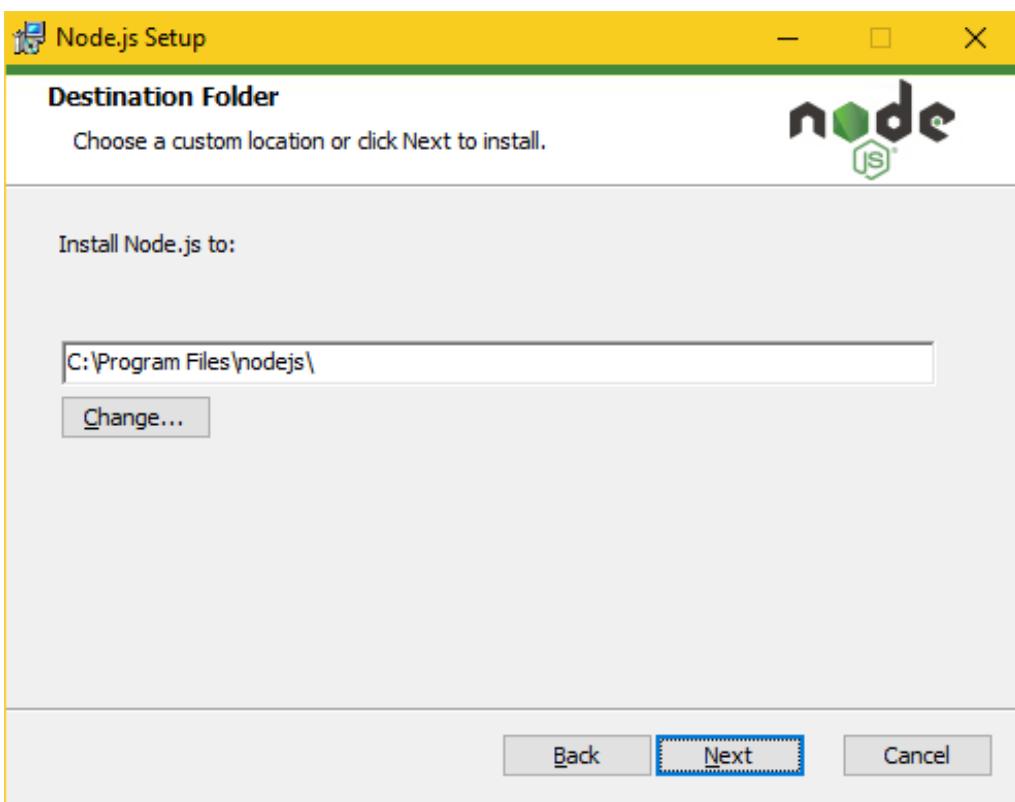
- Go to Downloads folder and double click on “node-v9.4.0-x64.msi”.



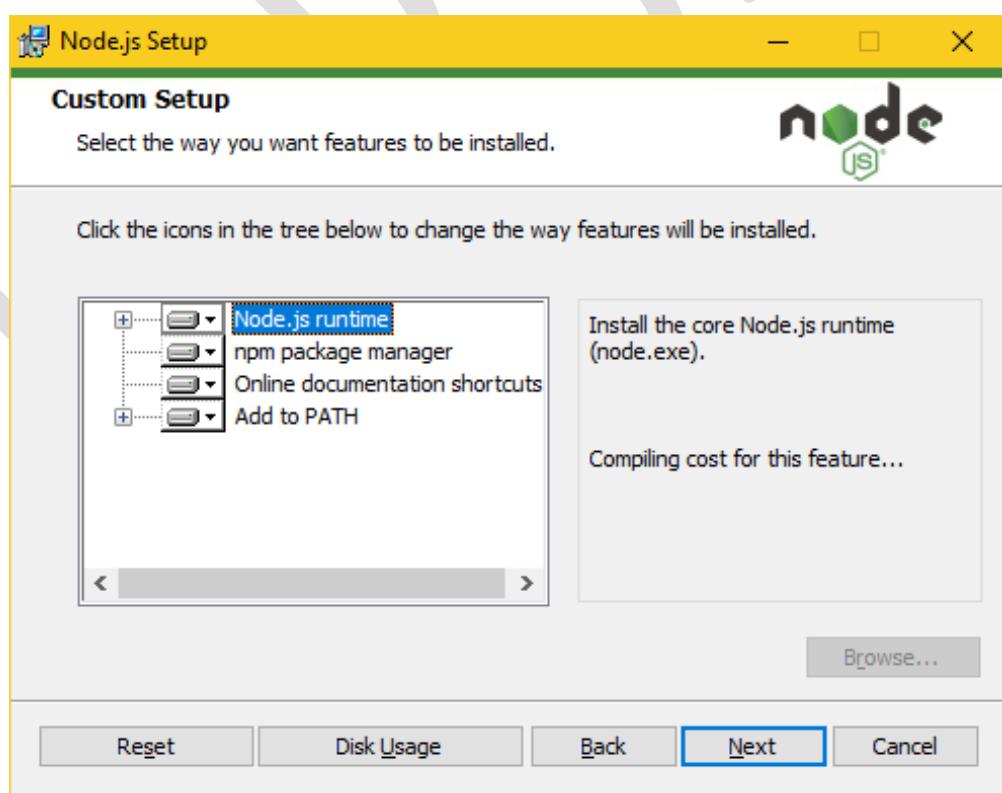
- Click on "Next".



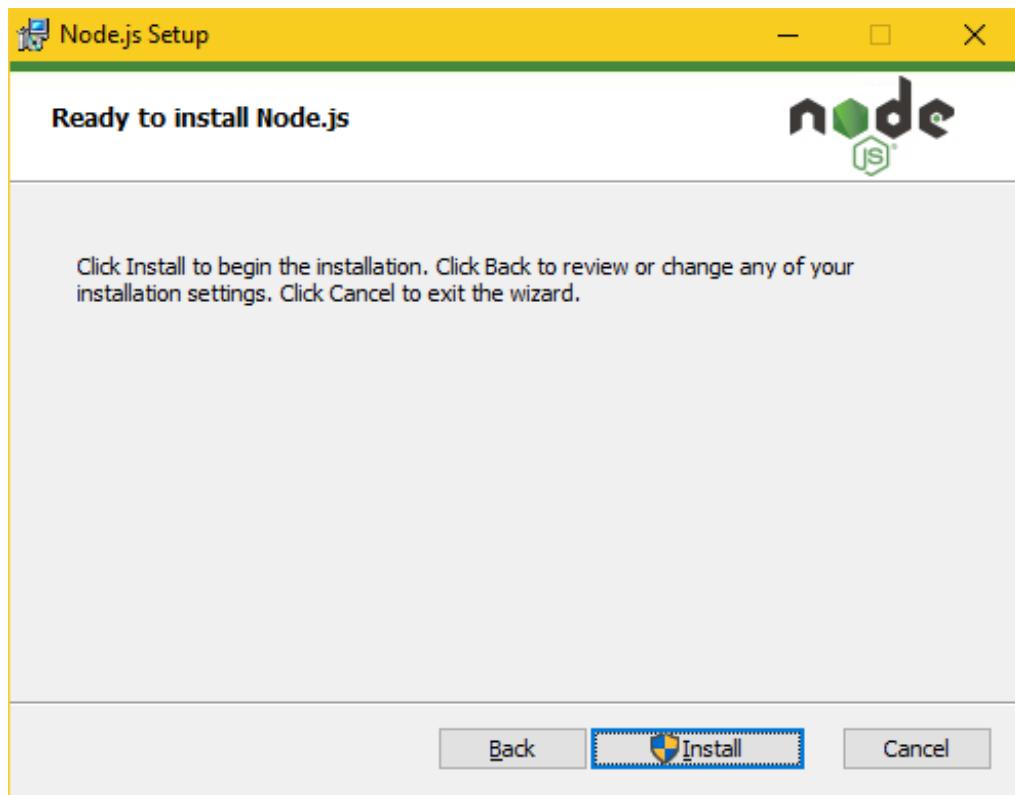
- Check the checkbox "I accept the terms in the License Agreement" and click on "Next".



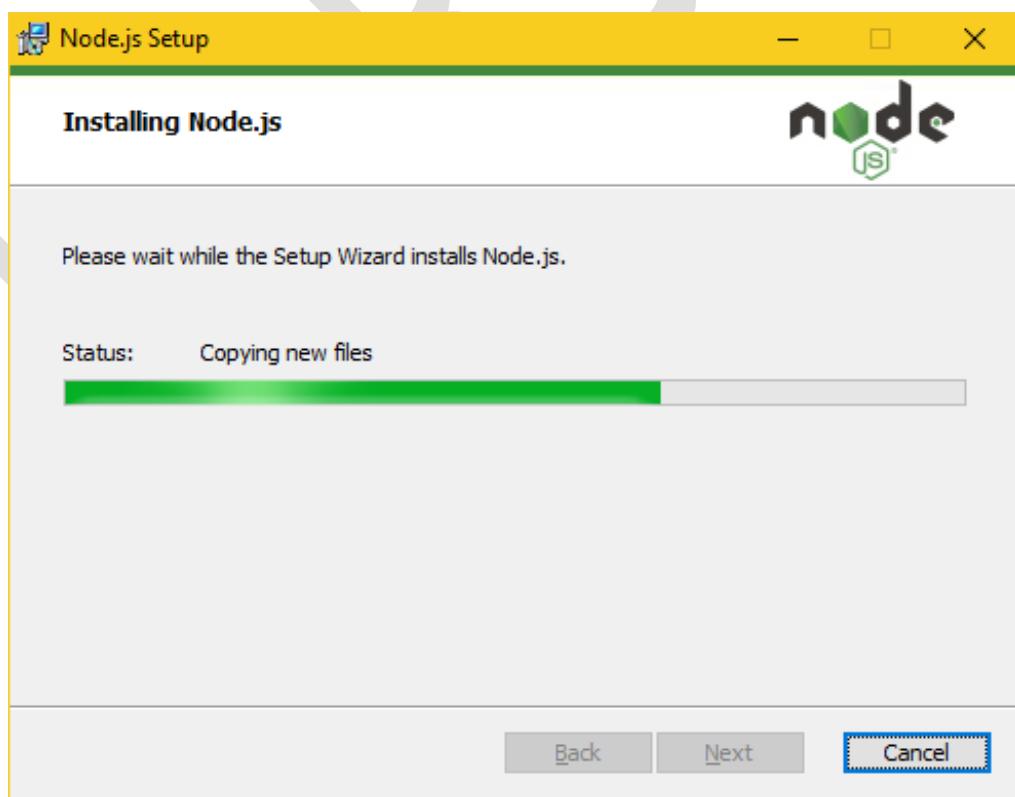
- Click on "Next".



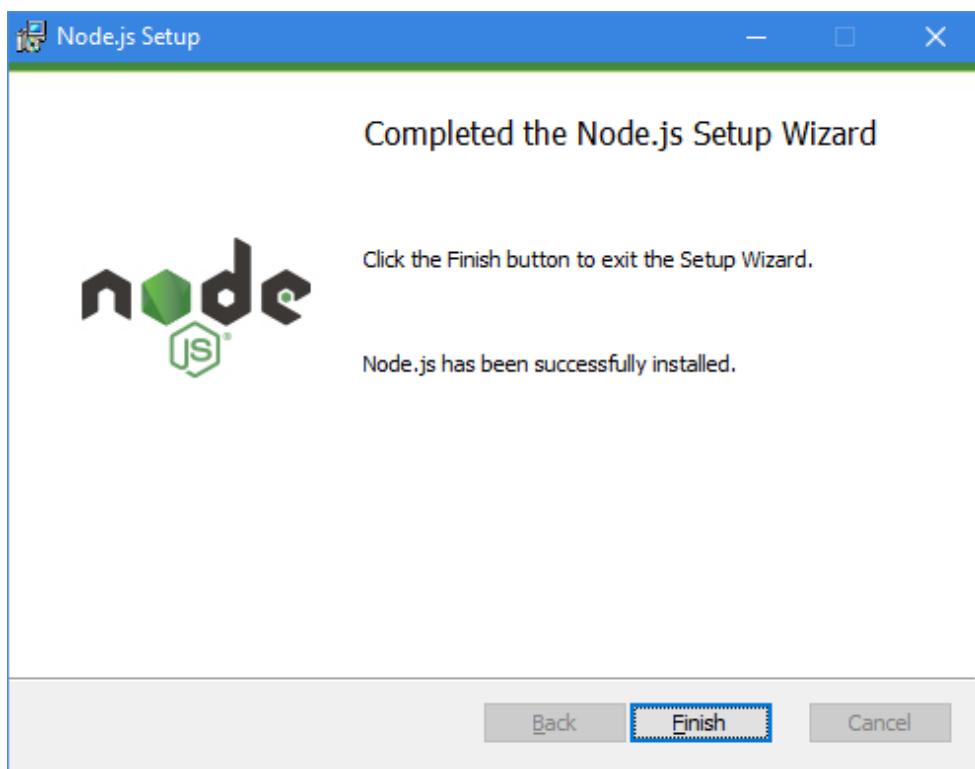
- Click on "Next".



- Click on “Install”.
- Click on “Yes”.



- Installation is in progress now.



- After installation is completed, click on "Finish". Now NodeJS installation is finished.
- **Note:** Automatically it adds the nodejs installation directory in the "Path" at environment variables of your computer.

2) Installing TypeScript

- Open "Command Prompt".
- Type **npm install typescript -g** in the Command prompt and press Enter.

A screenshot of a Windows Command Prompt window titled 'Command Prompt'. It shows the following text:

```
C:\> Microsoft Windows [Version 10.0.16299.334]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Harsha>npm install typescript -g
```

- After pressing Enter:

A screenshot of a Windows Command Prompt window titled 'Command Prompt'. It shows the following text:

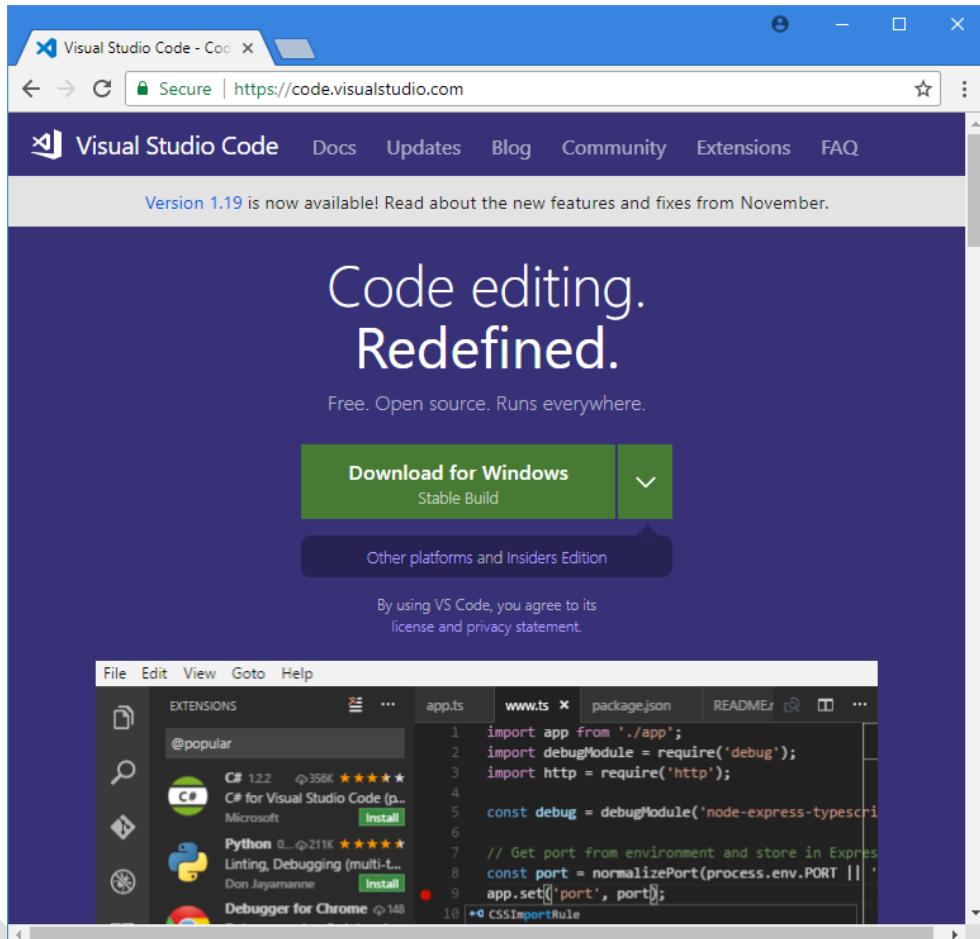
```
C:\> Command Prompt
C:\Users\Harsha>npm install typescript -g
C:\Users\Harsha\AppData\Roaming\npm\tsc -> C:\Users\Harsha\AppData\Roaming\npm\node_modules\typescript\bin\tsc
C:\Users\Harsha\AppData\Roaming\npm\tsserver -> C:\Users\Harsha\AppData\Roaming\npm\node_modules\typescript\bin\tsserver
+ typescript@2.8.1
added 1 package from 1 contributor in 1.902s

C:\Users\Harsha>
```

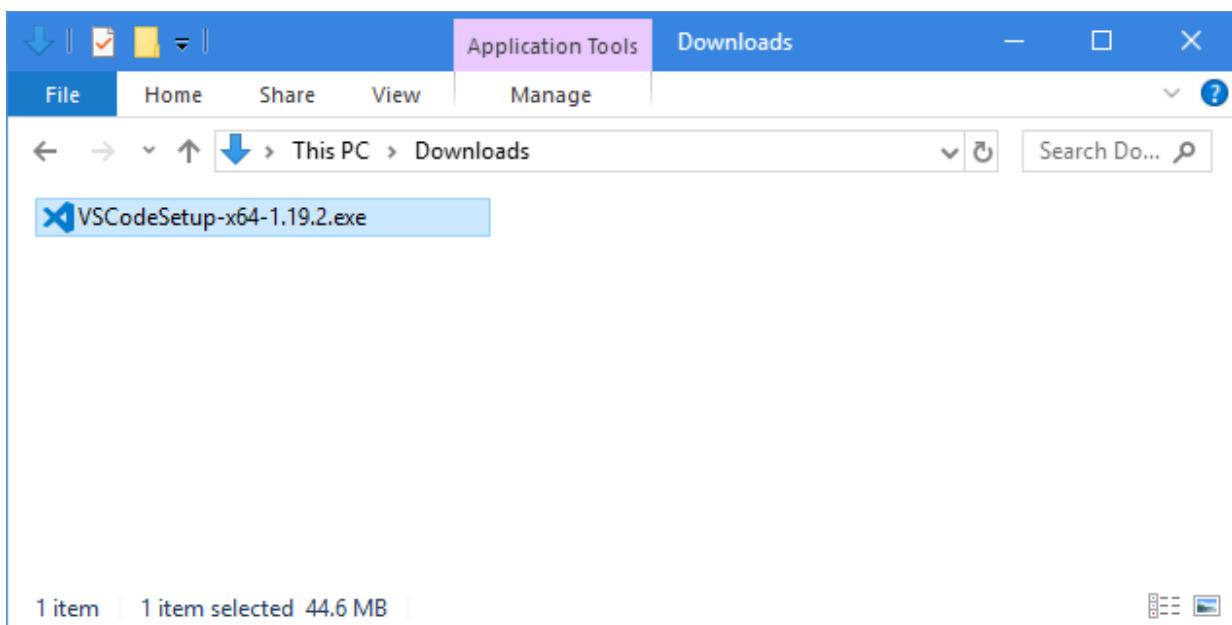
- Now TypeScript successfully installed.

3) Installing Visual Studio Code

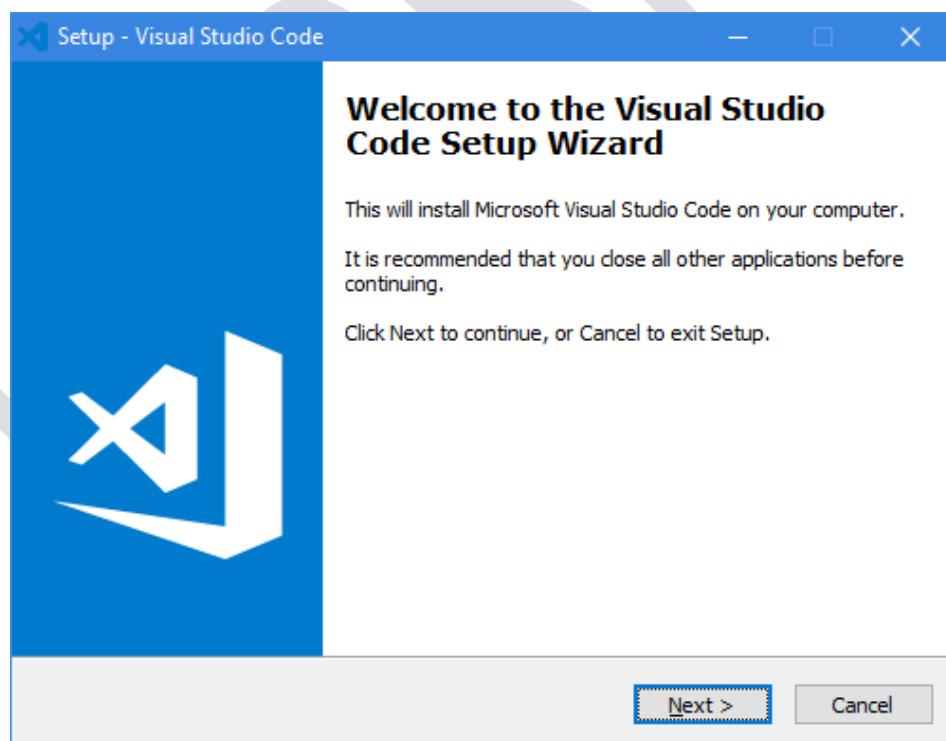
- If you have installed Visual Studio Code already, you can skip this step and go to step 4.
- If you don't have installed Visual Studio Code, continue this step.
- "Visual Studio Code" is the recommended editor for typescript and angular.
- Go to <https://code.visualstudio.com>



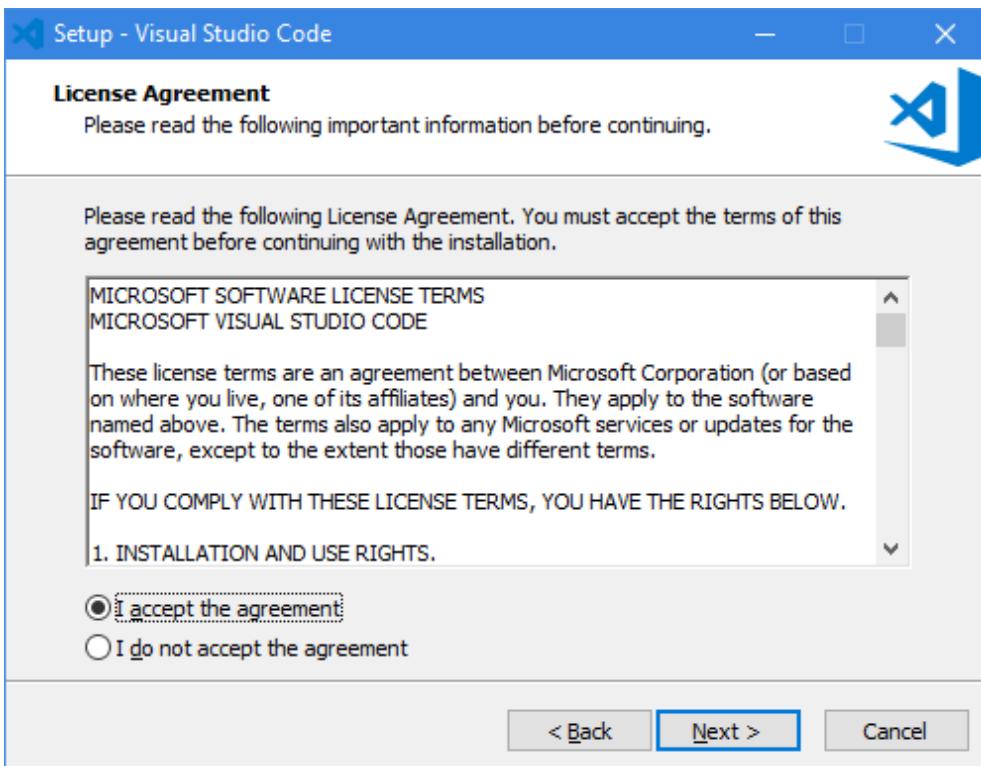
- Click on "Download for Windows".
- **Note:** The version number may be different at your practice time.
- Go to "Downloads" folder; you can find "VSCodeSetup-x64-1.19.2.exe" file.



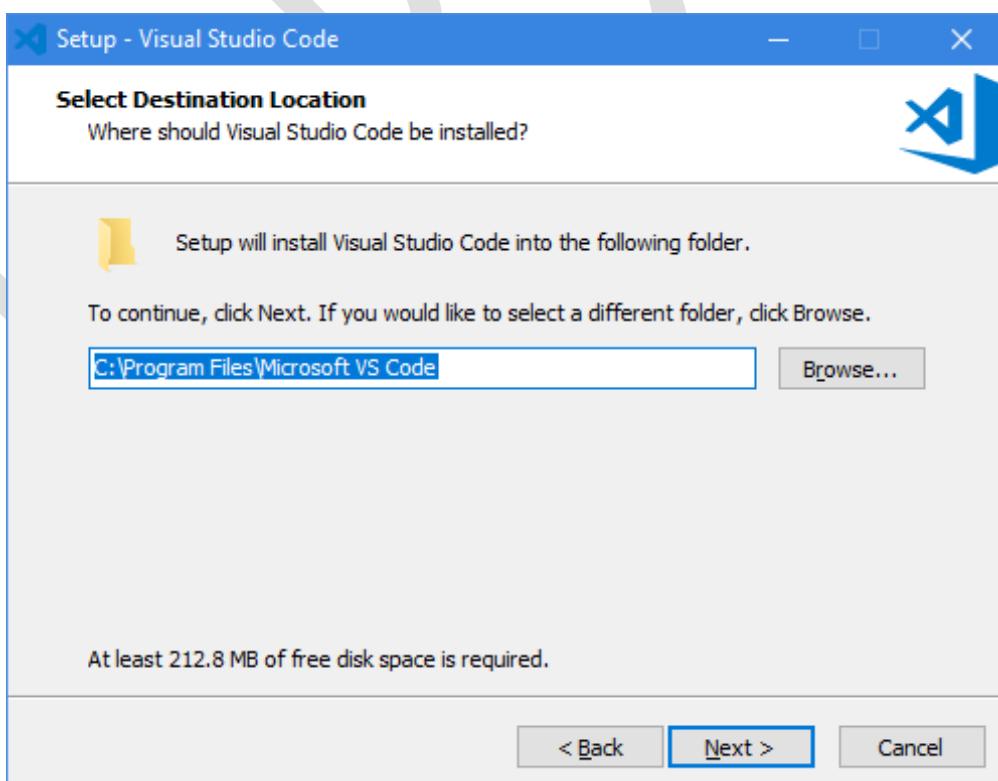
- Double click on “VSCodeSetup-x64-1.19.2.exe” file.
- Click on “Yes”.



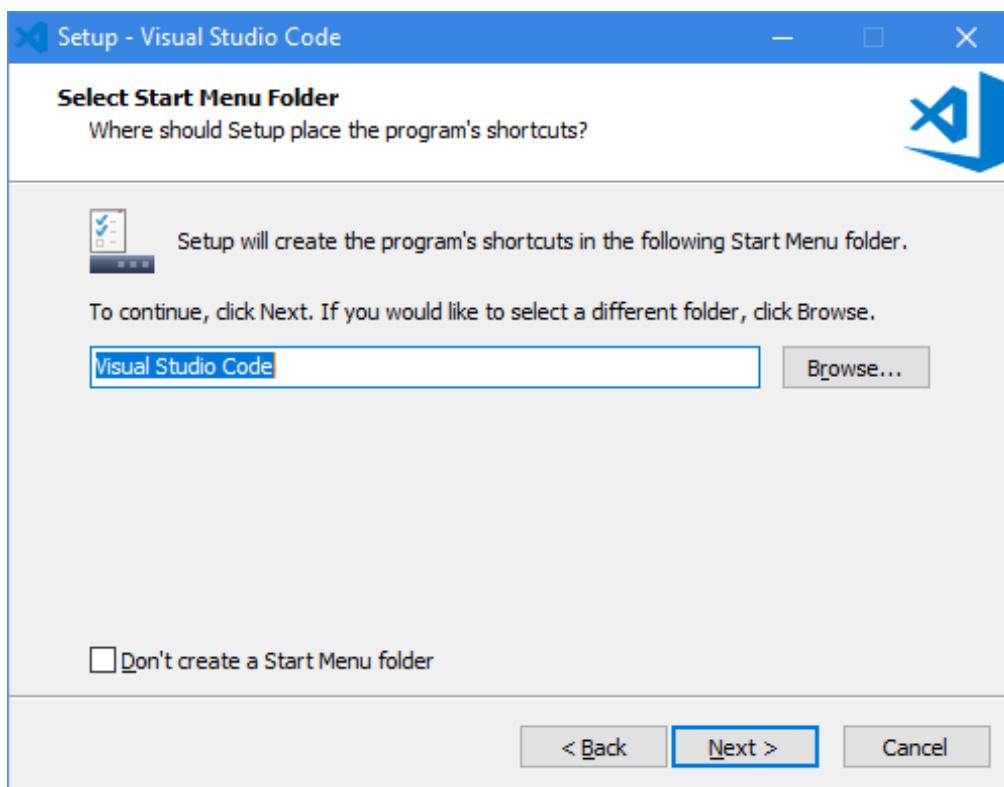
- Click on “Next”.



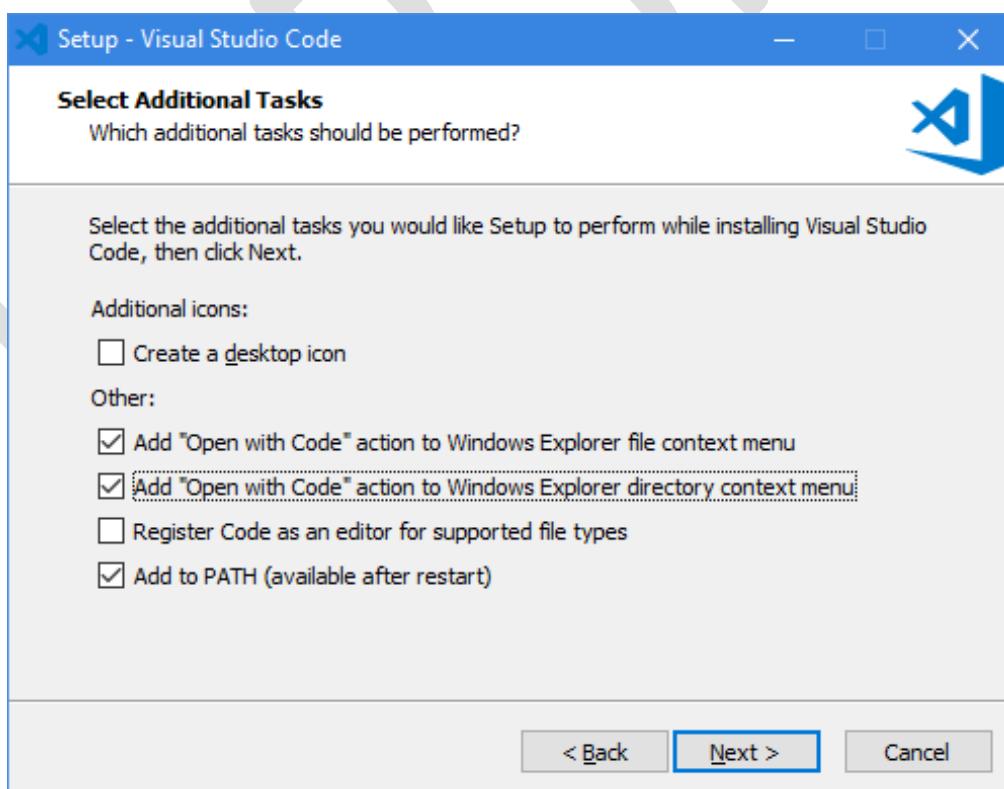
- Click on “I accept the agreement”.
- Click on “Next”.



- Click on “Next”.

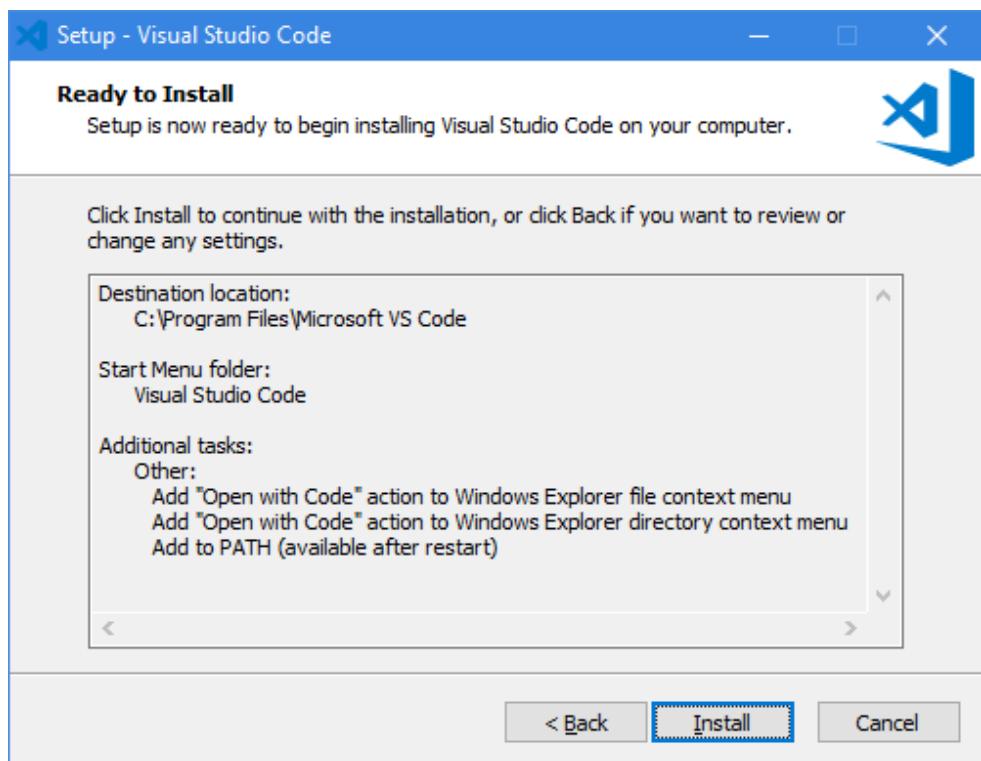


- Click on “Next”.

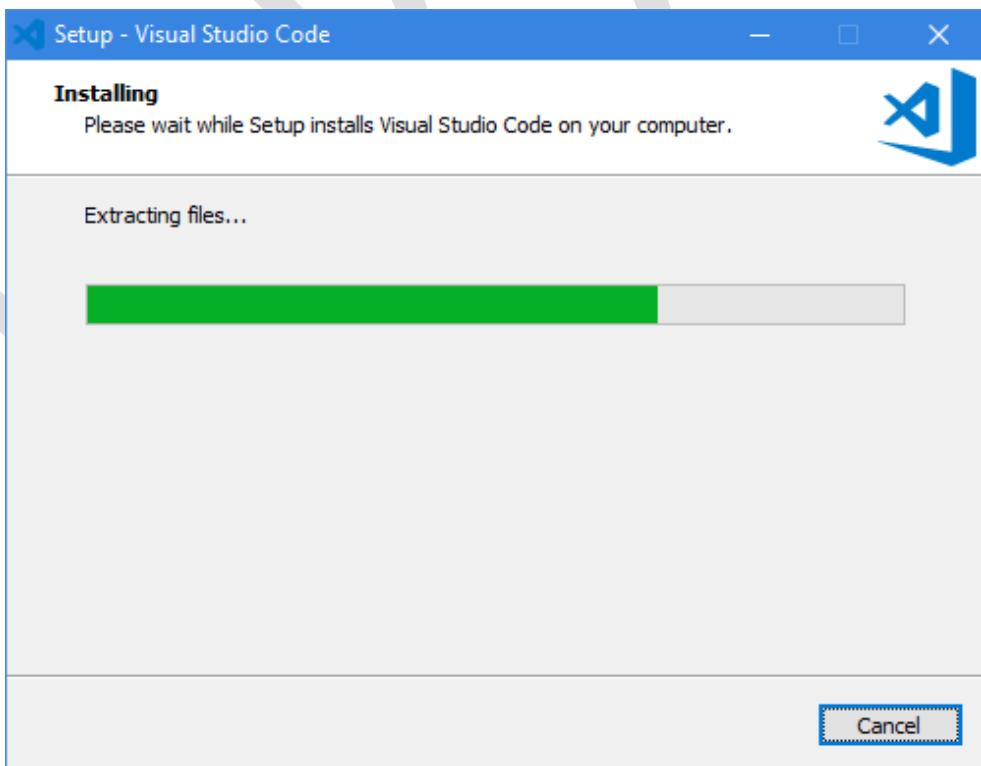


- Check the checkbox “Add Open with Code action to Windows Explorer file context menu”.
- Check the checkbox “Add Open with Code action to Windows Explorer directory context menu”.

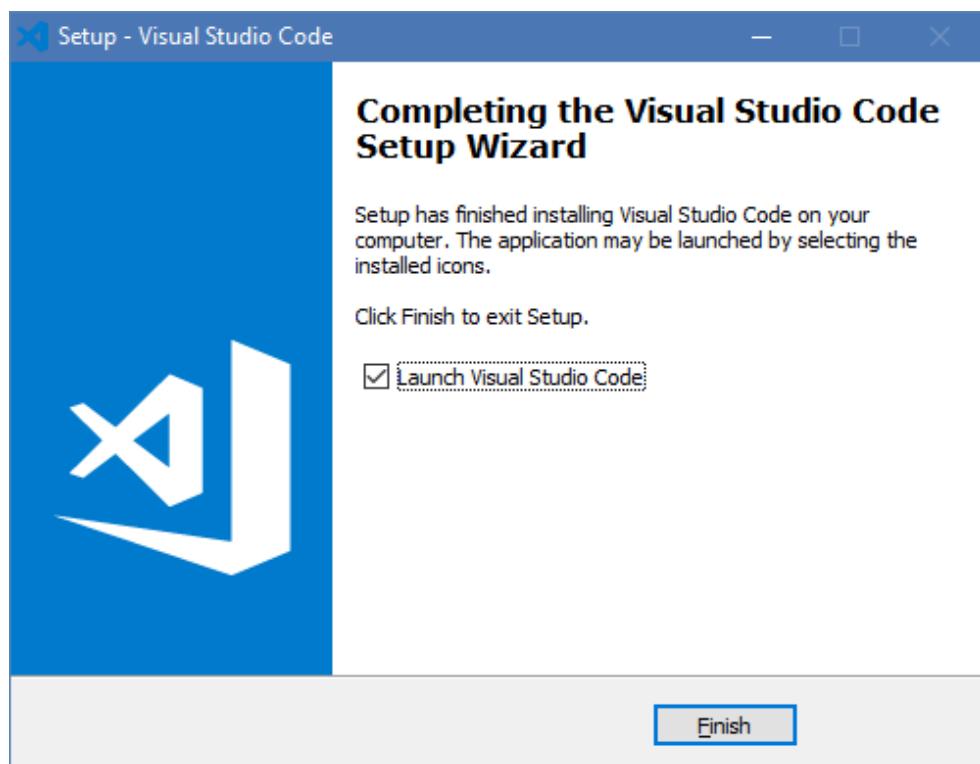
- Click on “Next”.



- Click on “Install”.



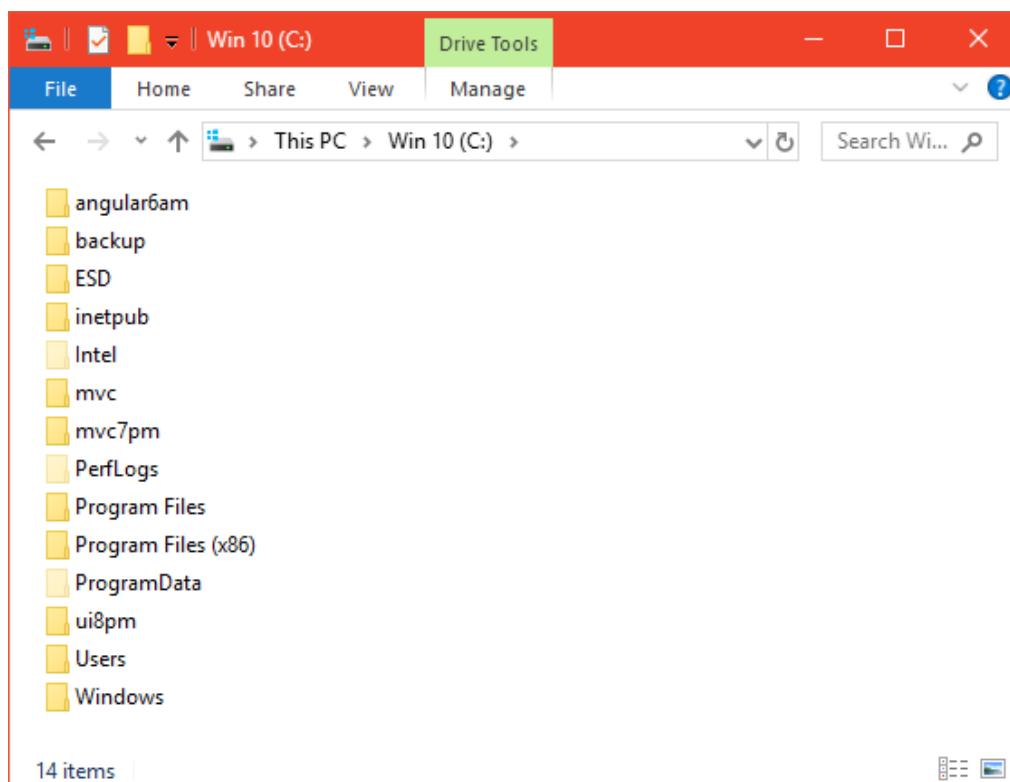
- Installation is going on....



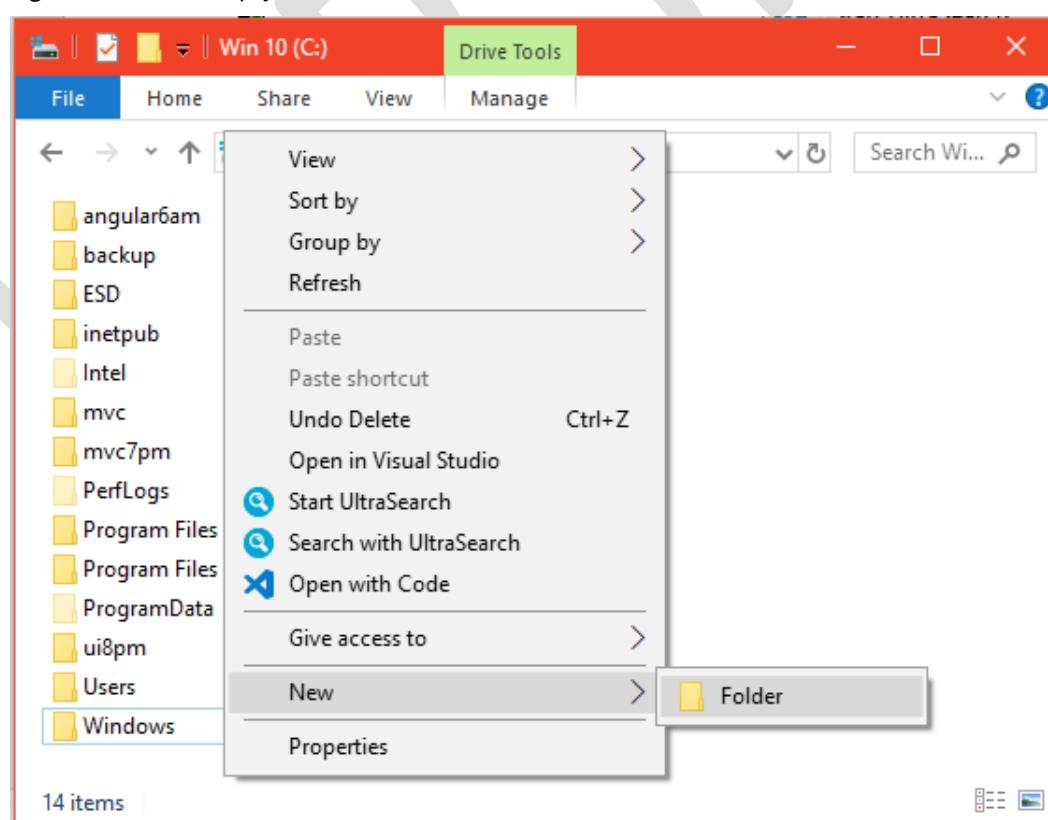
- Click on “Finish”.

4) Create Application Folder

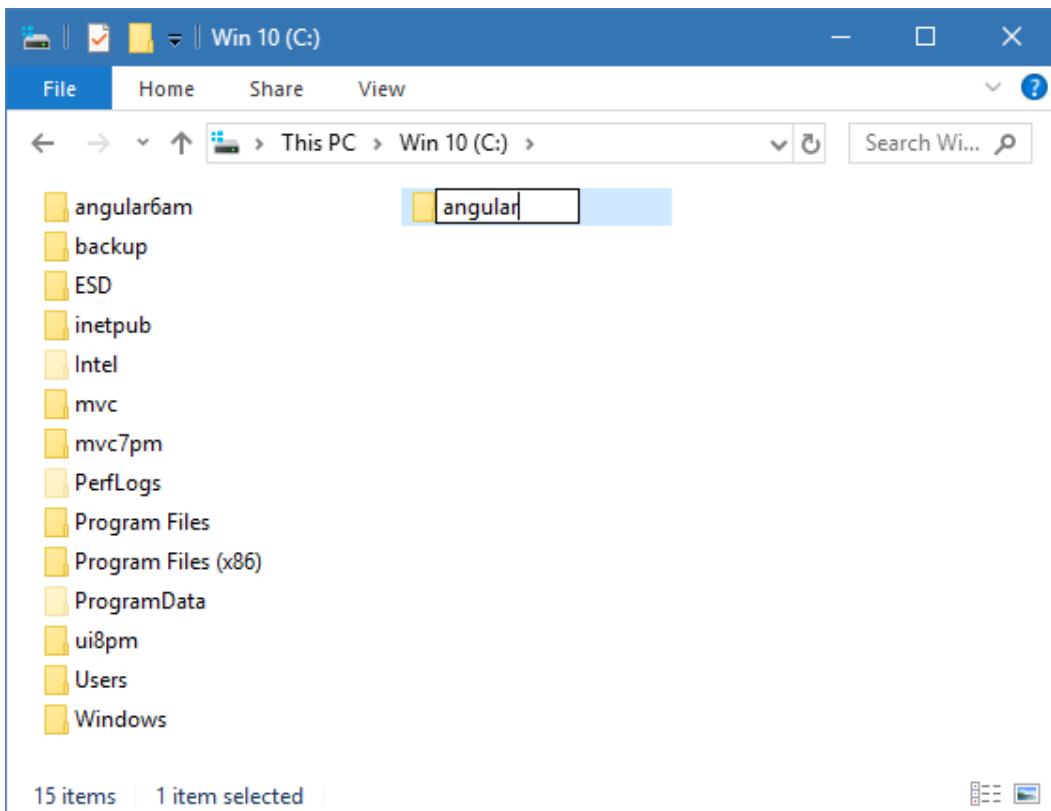
- If you have already created “c:\angular” folder, you can skip this step and go to step 5.
- If you don’t have created “c:\angular” folder, continue this step.
- Go to “Computer” (or) “This PC”.
- Go to “c:\”.



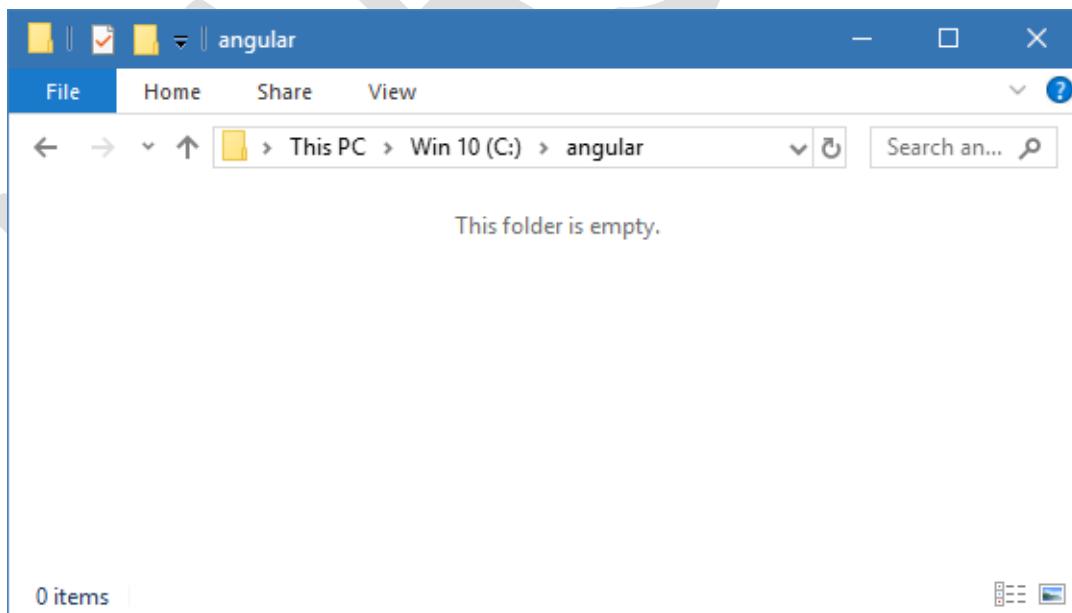
- Right click on the empty area and click on “New Folder”.



- Type the folder name as “angular” and press Enter.



- The “c:\angular” folder is ready now.



5) Install @angular/cli package

- Open “Command Prompt” and run the following command:

```
cd c:\angular
```

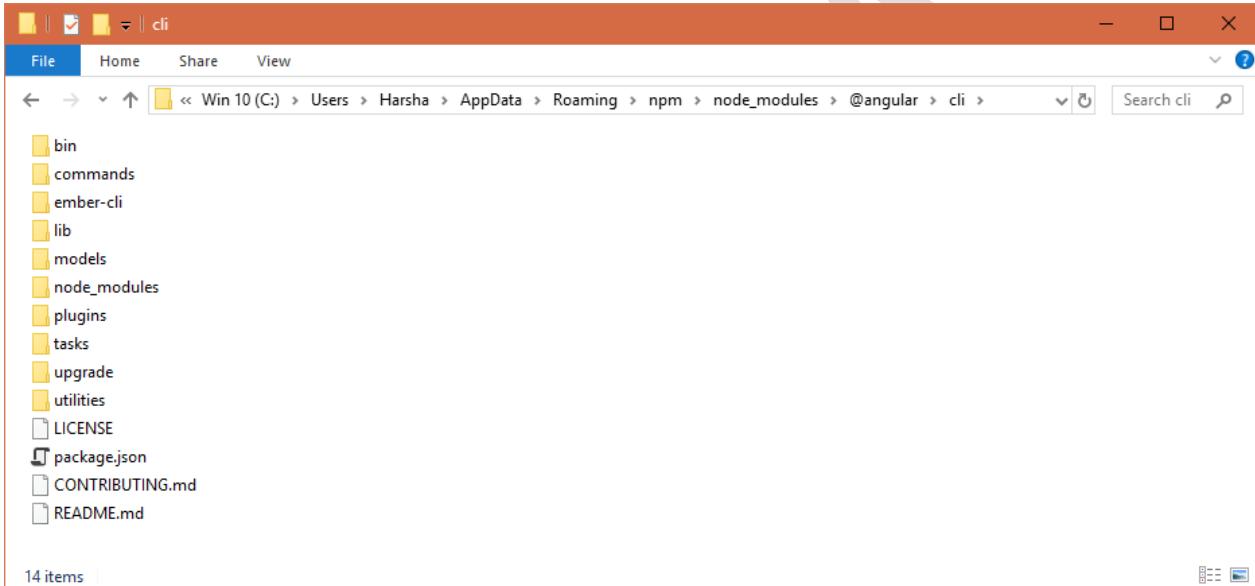
```
npm install @angular/cli -g
```



```
Command Prompt
Microsoft Windows [Version 10.0.16299.334]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Harsha>cd c:\angular
c:\angular>npm install @angular/cli -g
```

- The “@angular/cli” package provides a set of commands to create new angular applications and also to create items in the project such as modules, components, pipes, services, directives etc.
- Installing “@angular/cli” package globally will download the package files from internet into “C:\Users\Harsha\AppData\Roaming\npm\node_modules\@angular\cli” folder, which can be used from any folder in the entire system.



6) Creating New Angular Application

- Open “Command Prompt” and run the following command:

```
cd c:\angular
ng new app1
```



```
Command Prompt
Microsoft Windows [Version 10.0.16299.334]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Harsha>cd c:\angular
c:\angular>ng new app1
```

- After creating application:

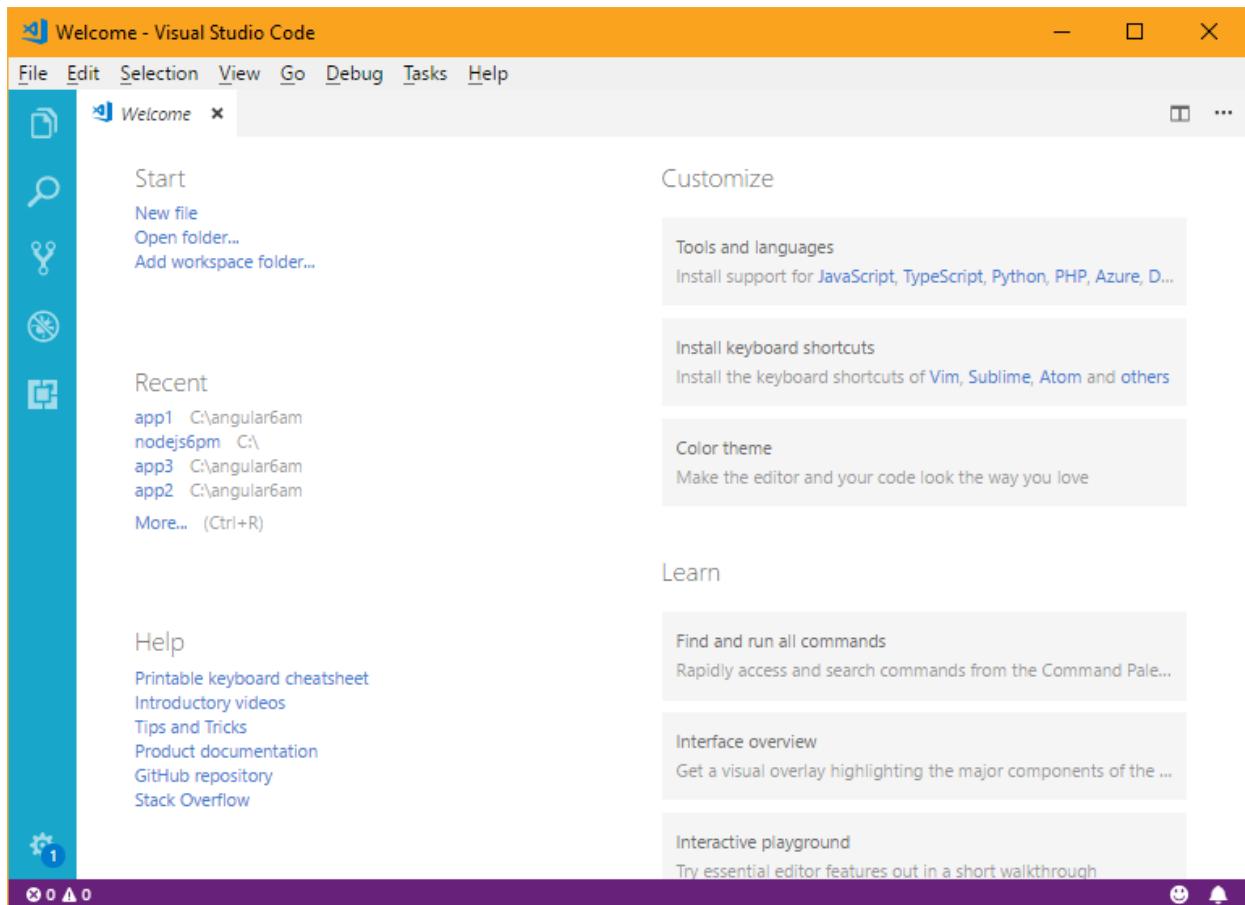


```
Command Prompt
2","arch":"x64"})  
added 1267 packages from 1240 contributors in 197.516s  
'git' is not recognized as an internal or external command,  
operable program or batch file.  
Project 'app1' successfully created.  
c:\angular>
```

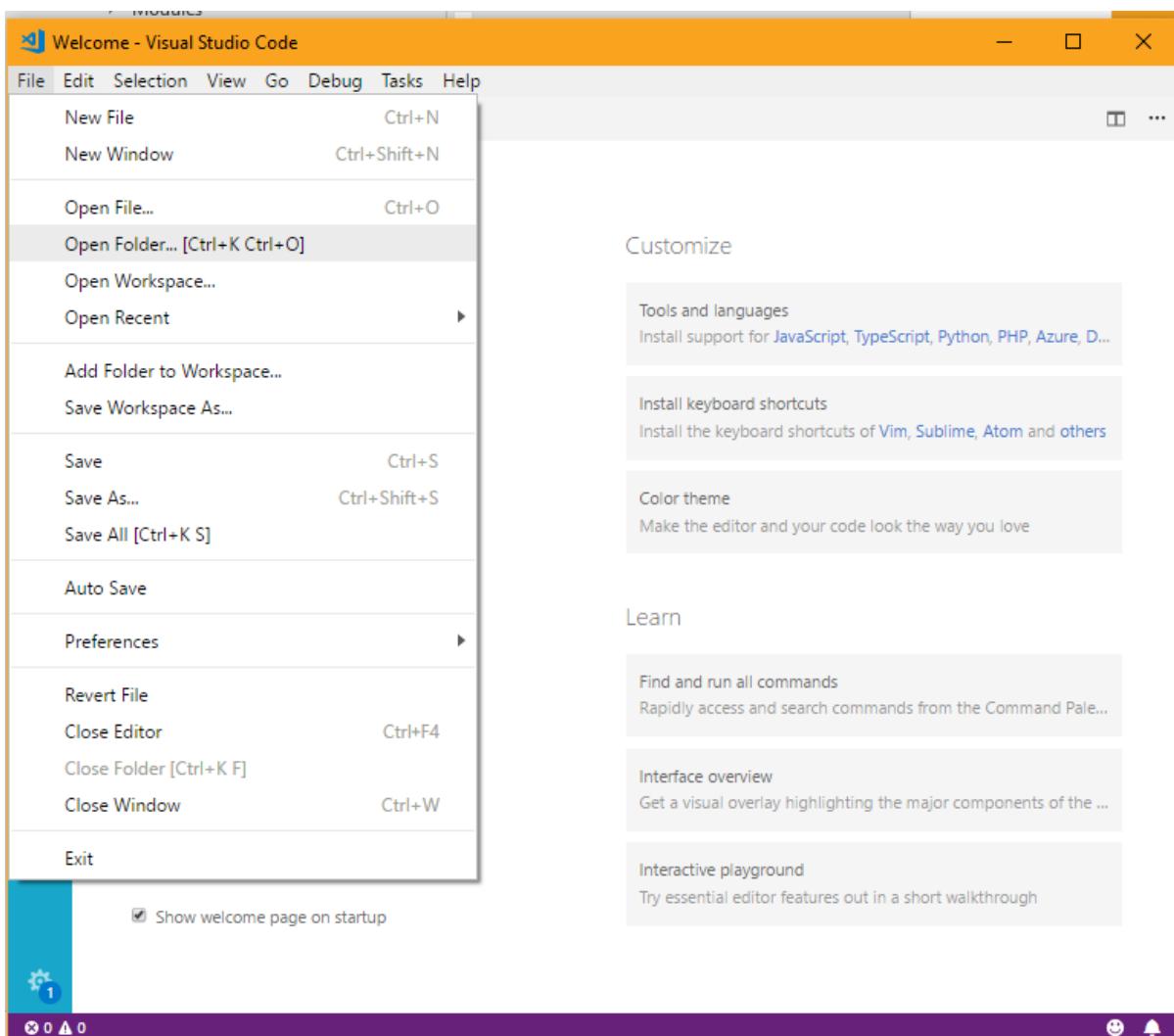
- Now new application at “c:\angular\app1” folder is created.

7) Open the Application in Visual Studio Code

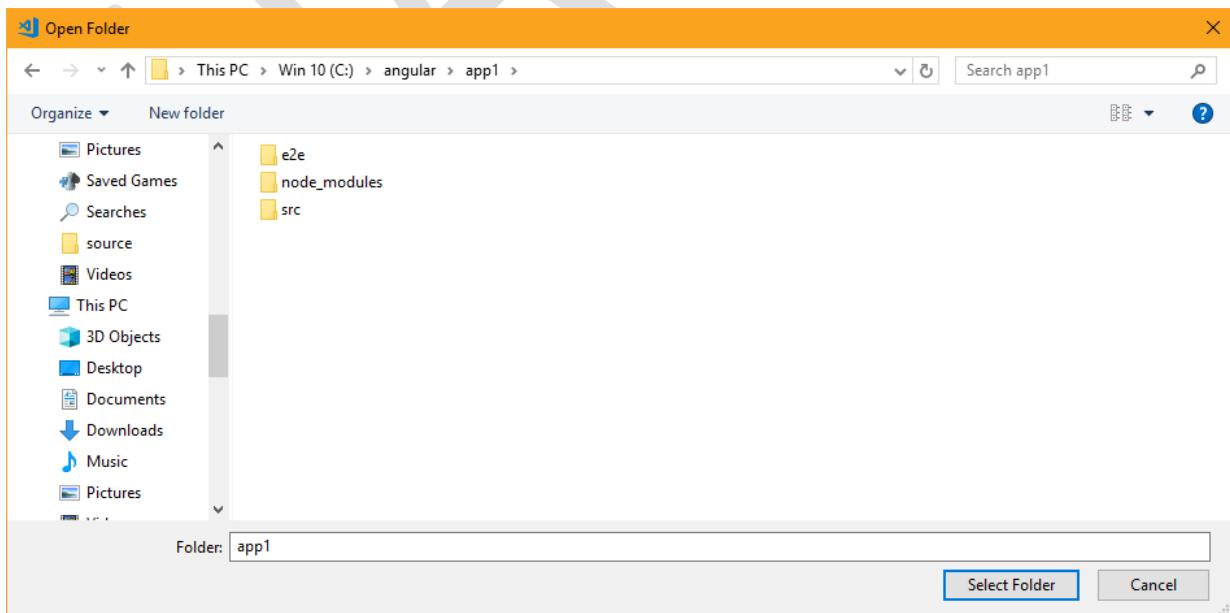
- Go to “Start” > “Visual Studio Code”.



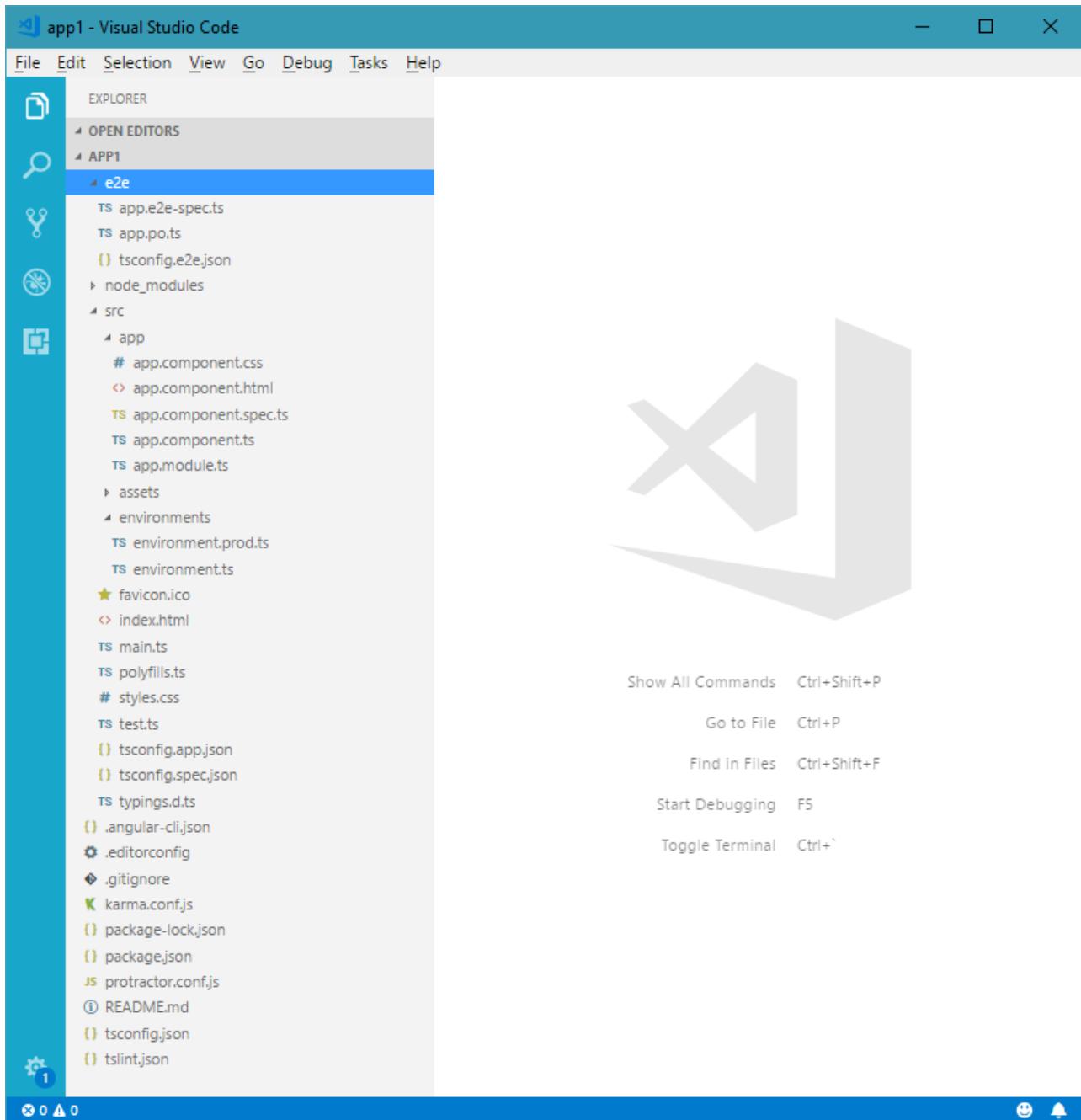
- Click on “File” – “Open Folder”.



- Select the folder "c:\angular\app1".



- Click on “Select Folder”.



8) app.component.html

- Go to “app1\src\app1\app.component.ts” in “Explorer” window in Visual Studio Code.

The screenshot shows the Visual Studio Code interface. The left sidebar displays the project structure under 'EXPLORER'. The main editor window shows the file 'app.component.html' with the content: '<div>Hello World</div>'. The status bar at the bottom indicates 'Ln 4, Col 1' and other settings.

```
<div>
  Hello World
</div>
```

9) Compile the application

- Open “Command Prompt” and run the following command:

```
cd c:\angular\app1
ng serve
```

The screenshot shows a Windows Command Prompt window. The user has navigated to the directory 'c:\angular\app1' and is running the command 'ng serve'. The output shows the Angular live development server starting up, indicating the application is compiled successfully.

- Application compiled successfully:

The screenshot shows the output of the 'ng serve' command. It includes the server listening message, build statistics for different bundles, and a confirmation that webpack was compiled successfully.

```
c:\angular>cd c:\angular\app1
c:\angular\app1>ng serve
** NG Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
Date: 2018-04-08T09:28:12.119Z
Hash: 2e719c08bd2b1732931f
Time: 9098ms
chunk {inline} inline.bundle.js (inline) 3.85 kB [entry] [rendered]
chunk {main} main.bundle.js (main) 15.2 kB [initial] [rendered]
chunk {polyfills} polyfills.bundle.js (polyfills) 554 kB [initial] [rendered]
chunk {styles} styles.bundle.js (styles) 41.5 kB [initial] [rendered]
chunk {vendor} vendor.bundle.js (vendor) 7.42 MB [initial] [rendered]

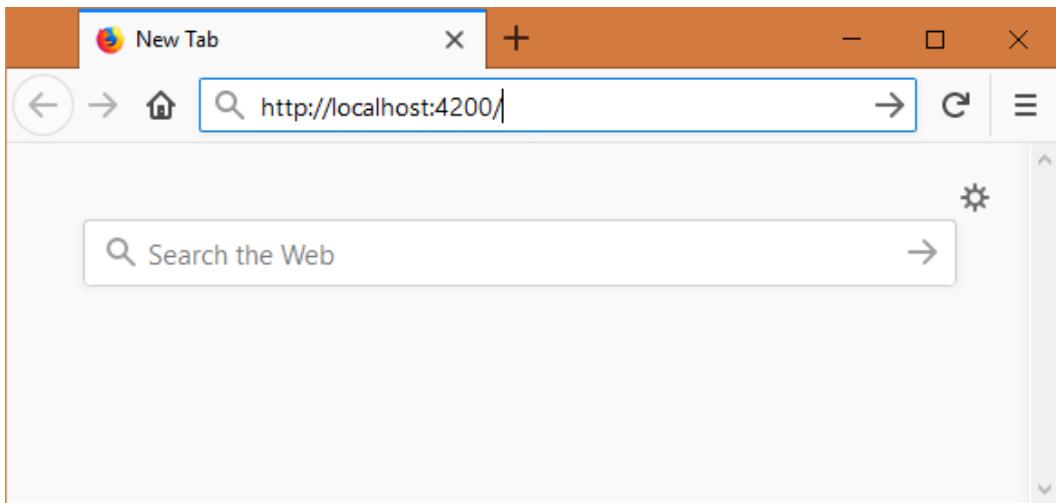
webpack: Compiled successfully.
```

- Now new application at “c:\angular\app1” folder is created.

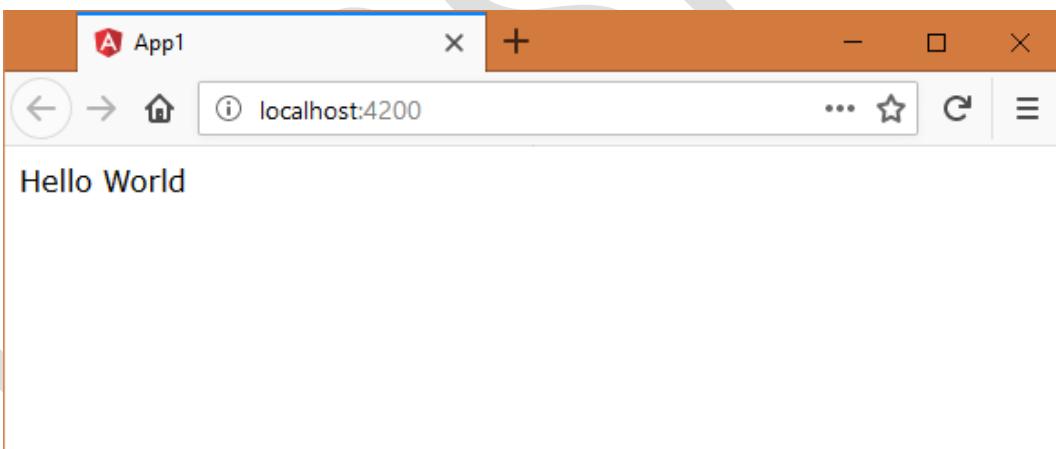
10) Run the application

- Open the browser and enter the following URL:

http://localhost:4200



- Output:



Folder Structure of Angular Application

- c:\angular (folder)
 - app1 (folder)
 - package.json
 - node_modules (folder)
 - package-lock.json
 - tsconfig.json
 - tslint.json
 - protractor.conf.js
 - karma.conf.js
 - angular.json

- src (folder)
 - polyfills.ts
 - favicon.ico
 - index.html
 - styles.css
 - main.ts
 - environments (folder)
 - assets (folder)
 - app (folders)
 - app.module.ts
 - app.component.ts
 - app.component.html
 - app.component.css
 - app.component.spec.ts

1) package.json

- The “package.json” file represents the configuration settings / meta data of the application.
- It specifies package name, version, dependencies etc.
- It is a fixed filename.
- It is must, without which the application is not accepted.
- It is a JSON files, which means it contains key/value pairs. Every key and value must be within double quotes (“ ”) / single quotes (‘ ’).

Properties of “package.json”

- | | |
|-------------------------|--|
| 1 <u>name</u> | Represents name of the application. It can be maximum of 214 characters. Non-URL friendly characters such as /, :, @ etc., are not allowed. <u>Ex:</u> “name”: “app1” |
| 2 <u>version</u> | Represents version of the application. <u>Ex:</u> 1.0.0 It should have 3 numbers major version, minor version, subminor version. <u>Ex:</u> “version”: “1.0.0” |
| 4 <u>license</u> | Represents license of the application. Ex: MIT, ISC MIT: MIT stands for “Massachusetts Institute of Technology”. |

MIT license allows to create private applications that can be used either privately within the organization and also can be shared with other known organizations.

Ex: “license”: “MIT”

ISC:

ISC stands for “Internet Systems Consortium”.

ISC license allows to create public applications that can be used anywhere.

Ex: “license”: “ISC”

5 scripts

Represents a set of commands that can run on the Command Prompt to run, test the applications using commands.

Ex: “scripts”:

```
{  
  "start": "ng serve"  
}
```

4 private

Represents whether the application should be used privately within the same organization or not.

If it is true, it can be used only within the same organization. Outside usage is not permitted.

Ex: “private”: true

5 dependencies

Represents the list of packages that are to be installed to run the application. These packages will be installed in both developer machine and production server.

Ex: “dependencies”:

```
{  
  "@angular/core": "^5.2.0"  
}
```

6 devDependencies

Represents the list of packages that are to be installed to develop the application. These packages will be installed only in the developer machine; not in the production server.

Ex: “devDependencies”:

```
{  
  "@angular/cli": "~1.7.4"  
}
```

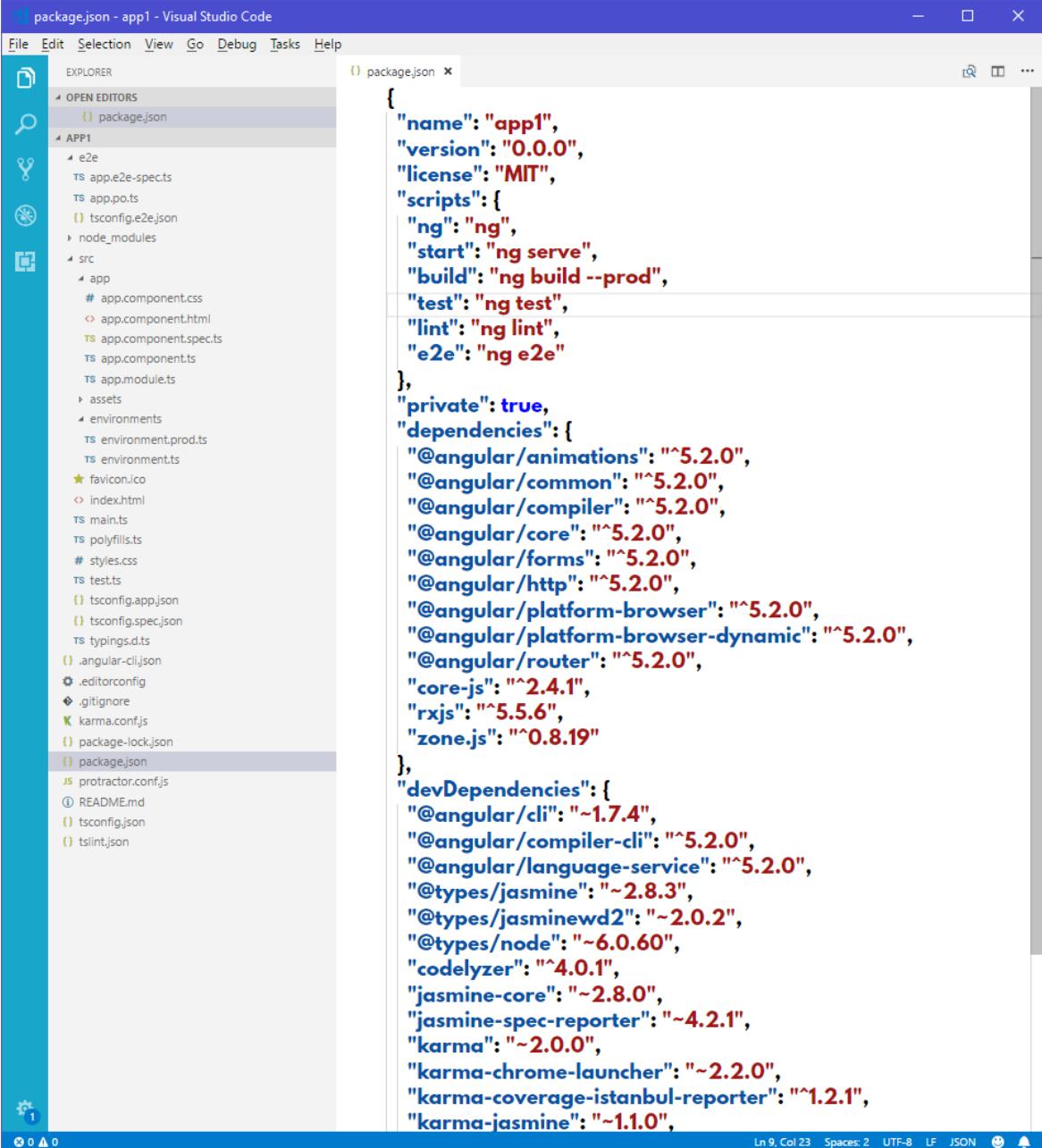
package.json

```
{  
  "name": "app1",  
  "version": "0.0.0",  
  "license": "MIT",  
  "scripts": {  
    "ng": "ng",  
    "start": "ng serve",  
    "build": "ng build --prod",  
    "test": "ng test",  
    "lint": "ng lint",  
    "e2e": "ng e2e"  
  },  
  "private": true,  
  "dependencies": {  
    "@angular/animations": "^5.2.0",  
    "@angular/common": "^5.2.0",  
    "@angular/compiler": "^5.2.0",  
    "@angular/core": "^5.2.0",  
    "@angular/forms": "^5.2.0",  
    "@angular/http": "^5.2.0",  
    "@angular/platform-browser": "^5.2.0",  
    "@angular/platform-browser-dynamic": "^5.2.0",  
    "@angular/router": "^5.2.0",  
    "core-js": "^2.4.1",  
    "rxjs": "^5.5.6",  
    "zone.js": "^0.8.19"  
  },  
  "devDependencies": {  
    "@angular/cli": "~1.7.4",  
    "@angular/compiler-cli": "^5.2.0",  
    "@angular/language-service": "^5.2.0",  
    "@types/jasmine": "~2.8.3",  
    "@types/jasminewd2": "~2.0.2",  
    "@types/node": "~6.0.60",  
    "codelyzer": "^4.0.1",  
    "jasmine-core": "~2.8.0",  
    "jasmine-spec-reporter": "~4.2.1",  
    "karma": "~2.0.0",  
    "karma-chrome-launcher": "~2.2.0",  
    "karma-coverage-istanbul-reporter": "^1.2.1",  
    "karma-jasmine": "~1.1.0",  
  }  
}
```

```

    "karma-jasmine-html-reporter": "^0.2.2",
    "protractor": "~5.1.2",
    "ts-node": "~4.1.0",
    "tslint": "~5.9.1",
    "typescript": "~2.5.3"
  }
}

```



```

{
  "name": "app1",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build --prod",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^5.2.0",
    "@angular/common": "^5.2.0",
    "@angular/compiler": "^5.2.0",
    "@angular/core": "^5.2.0",
    "@angular/forms": "^5.2.0",
    "@angular/http": "^5.2.0",
    "@angular/platform-browser": "^5.2.0",
    "@angular/platform-browser-dynamic": "^5.2.0",
    "@angular/router": "^5.2.0",
    "core-js": "^2.4.1",
    "rxjs": "^5.5.6",
    "zone.js": "^0.8.19"
  },
  "devDependencies": {
    "@angular/cli": "~1.7.4",
    "@angular/compiler-cli": "^5.2.0",
    "@angular/language-service": "^5.2.0",
    "@types/jasmine": "~2.8.3",
    "@types/jasminewd2": "~2.0.2",
    "@types/node": "~6.0.60",
    "codelyzer": "^4.0.1",
    "jasmine-core": "~2.8.0",
    "jasmine-spec-reporter": "~4.2.1",
    "karma": "~2.0.0",
    "karma-chrome-launcher": "~2.2.0",
    "karma-coverage-istanbul-reporter": "~1.2.1",
    "karma-jasmine": "~1.1.0",
    "protractor": "~5.1.2"
  }
}

```

Packages of Angular 2+

- The angular 2+ framework is available as a collection of packages.
- Each feature of the framework is represented as a package. For example, routing is available as a package called “@angular/router”.
- We have to install (download) those packages in order to develop angular application.
- The packages of angular 2+ are divided into two types:

I. Angular Packages:

- These packages are part of angular 2+ framework.

II. Non-Angular Packages:

- These packages are not part of angular 2+ framework; provided by third party companies, but needed / useful in angular applications.

I. Angular Packages:

1 @angular/core

This package provides classes and interfaces that are related to decorators, component life cycle, dependency injection etc., that are needed in every angular 2+ application.

This is the mandatory package.

Examples of decorators provided by @angular/core package: @Component, @NgModule, @Input, @Injectable, @Inject etc.

Examples of component life cycle interfaces provided by @angular/core package: OnInit, DoCheck, AfterViewChecked, OnDestroy etc.

This package contains a module called “ApplicationModule”, which contains the set of pre-defined scripts that are needed to run the angular application.

2 @angular/common

This package provides common directives and pipes that are needed in most of the angular applications.

This is the mandatory package.

Ex of directives provided by @angular/common package: ngIf, ngFor, ngSwitch, ngClass etc.

Ex of pipes provided by @angular/common package: uppercase, lowercase, date, currency etc.

This package contains a module called “CommonModule”, which contains the above specified directives and pipes.

3 @angular/platform-browser

This package imports “ApplicationModule” from “@angular/core”, “CommonModule” from “@angular/common” and re-exports them as “BrowserModule”. Additionally, it provides some runtime services (such as “error handling, history handling” etc.), that are needed while running the application in the browser.

This is the mandatory package.

4 @angular/compiler

This package is used to compile the “template” into a “javascript code”. We never invoke the angular compiler directly; we will call it indirectly through either “@angular/platform-browser-dynamic” or “@angular/platform-browser”.

This is the mandatory package.

5 @angular/platform-browser-dynamic

An angular application can have any no. of modules. This package is used to bootstrap (start) executing a module, which execution should be started automatically at application startup.

This is the mandatory package.

6 @angular/forms

This package is used for creating “two way data bindings” and “validations” in angular 2+ applications. This package works based on another package called “zone.js”.

This package has two modules: FormsModule and ReactiveFormsModule.

This is the optional package.

7 @angular/router

This package is used to creating “routing” (page navigation) in angular 2+ applications.

This package has one module: RouterModule

This is the optional package.

8 @angular/http

This package is used to send ajax requests to server and get ajax response from server. This package works based on another package called “rxjs”.

This package has one module: HttpClientModule

This is the optional package.

9 @angular/animations

This package is used to create animations in angular 2+ applications.

This package has one module: AnimationsModule

This is the optional package.

10 @angular/material

Used to use “angular material design” in angular applications.

This package has several modules: MatButtonModule, MatCheckboxModule, MatMenuModule etc.

This is the optional package.

11 @angular/cdk

Based on this package only, “angular material design” components are developed. So this package must be used while using “@angular/material” package.

This is the optional package.

12 @angular/cli

This package provides a set of commands to create new angular application and its code items such as components, pipes, directives, services etc.

This is the mandatory package.

This package must be installed globally, with “-g” option.

II. Non-Angular Packages

1 typescript

This package provides typescript compiler, which is used to compile “typescript files (.ts)” into “javascript files (.js)”.

This is the mandatory package.

This package must be installed globally, with “-g” option.

2 systemjs

This package is used to load both angular framework-related and application program-related “.js” files into the browser.

This is the mandatory package.

3 core-js

This package contains polyfills, which are needed to run angular 5 application in Internet Explorer 9+.

This is the mandatory package.

4 rxjs

This package “rxjs” (Reactive Extensions for JavaScript) provides necessary code for making ajax calls to server.

This is the mandatory package.

5 zone.js

This package identifies the events in the browser and informs the same to angular framework; so that it can perform “change detection”.

This is the mandatory package.

6 jasmine

This package is used to write test cases for unit testing of components.

This is the mandatory package.

7 jasmine

This package is used to execute test cases within one browser.

This is the mandatory package.

8 karma

This package is used to execute test cases on different browsers.

This is the mandatory package.

9 tslint

This package is used to check whether the typescript files are following set of rules or not. For example, you can check the maximum length of the line in the code.

This is the mandatory package.

2) **tsconfig.json**

- Every compiler has some configuration settings.
- This file is used to set configuration settings of the “tsc” (Type Script Compiler).
- The “tsc” compiler automatically reads the “tsconfig.json” file; and then only it compiles the “.ts” files into “.js” file.

- This is a fixed filename.

“tsconfig.json” – Configuration Settings

1 target

Represents javascript version, into which the “.ts” file have to be compiled. Ex: “es5”.

Recommended: “es5”.

Options: es3 | es5 | es6

2 sourceMap

true | false

true: Generates “source map” files. The source map file contains mapping between line numbers of “.ts” file to “.js” file. Based on the source map files, you can debug the typescript code. It is recommended to generate source map files during development.

false: “Source map” files will not be generated. So then we can debug “javascript code” only. Source map files are not required in production.

3 experimentalDecorators

true | false

true: It supports decorators (such as @Component, @NgModule, @Injectable etc.)

false: It doesn’t support decorators.

4 emitDecoratorsMetaData

true | false

true: It supports decorators with meta data.

false: It doesn’t support decorators with meta data.

5 lib

It represents the list of library files to be included while compilation of the typescript files.

The angular 2+ application requires the following libraries:

[“es2017”, “dom”]

Note: These libraries will be installed automatically, along with “typescript” package.

es2017:

This library contains essential data type such as Number, String, Boolean, Object, Function, RegExp etc.

dom:

This library contains essential classes such as “HTMLElement”, “Document”, “Window” etc.

6 outDir

Specifies the directory (folder) where the compiled files needs to be stored.

```
{  
  "compileOnSave": false,  
  "compilerOptions": {  
    "outDir": "./dist/out-tsc",  
    "sourceMap": true,  
    "declaration": false,  
    "moduleResolution": "node",  
    "emitDecoratorMetadata": true,  
    "experimentalDecorators": true,  
    "target": "es5",  
    "typeRoots": [  
      "node_modules/@types"  
    ],  
    "lib": [  
      "es2017",  
      "dom"  
    ]  
  }  
}
```

3) tslint.json

- This file contains configuration settings for “tslint” tool, which is used to verify whether the typescript files are following a set of coding standards or not.
- For example, we can check the maximum length of the line, indentation etc.

```
{  
  "rulesDirectory": [  
    "node_modules/codelyzer"  
  ],  
  "rules": {  
    "arrow-return-shorthand": true,  
    "callable-types": true,  
    "class-name": true,  
    "component-class-name": true,  
    "component-selector": true,  
    "directive-class-name": true,  
    "directive-selector": true,  
    "for-in-check": true,  
    "function-signature": true,  
    "interface-name": true,  
    "no-let": true,  
    "no-new-object": true,  
    "no-redundant-jsdoc": true,  
    "no-self-asynchronous": true,  
    "no-unbound-method": true,  
    "no-var": true,  
    "one-line": true,  
    "one-variable-per-declaration": true,  
    "prefer-const": true,  
    "semicolon": true,  
    "template-delimiter": true,  
    "use-pipe": true,  
    "use-type-guard-operands": true  
  }  
}
```

```
"class-name": true,  
"comment-format": [  
  true,  
  "check-space"  
],  
"curly": true,  
"deprecation": {  
  "severity": "warn"  
},  
"eofline": true,  
"forin": true,  
"import-blacklist": [  
  true,  
  "rxjs",  
  "rxjs/Rx"  
],  
"import-spacing": true,  
"indent": [  
  true,  
  "spaces"  
],  
"interface-over-type-literal": true,  
"label-position": true,  
"max-line-length": [  
  true,  
  140  
],  
"member-access": false,  
"member-ordering": [  
  true,  
  {  
    "order": [  
      "static-field",  
      "instance-field",  
      "static-method",  
      "instance-method"  
    ]  
  }  
],  
"no-arg": true,  
"no-bitwise": true,  
"no-console": [  
  true,  
  "debug",  
  "info",  
  "time",  
  "timeEnd",
```

```
"trace"
],
"no-construct": true,
"no-debugger": true,
"no-duplicate-super": true,
"no-empty": false,
"no-empty-interface": true,
"no-eval": true,
"no-inferable-types": [
  true,
  "ignore-params"
],
"no-misused-new": true,
"no-non-null-assertion": true,
"no-shadowed-variable": true,
"no-string-literal": false,
"no-string-throw": true,
"no-switch-case-fall-through": true,
"no-trailing-whitespace": true,
"no-unnecessary-initializer": true,
"no-unused-expression": true,
"no-use-before-declare": true,
"no-var-keyword": true,
"object-literal-sort-keys": false,
"one-line": [
  true,
  "check-open-brace",
  "check-catch",
  "check-else",
  "check-whitespace"
],
"prefer-const": true,
"quotemark": [
  true,
  "single"
],
"radix": true,
"semicolon": [
  true,
  "always"
],
"triple-equals": [
  true,
  "allow-null-check"
],
"typedef-whitespace": [
  true,
```

```
{  
  "call-signature": "nospace",  
  "index-signature": "nospace",  
  "parameter": "nospace",  
  "property-declaration": "nospace",  
  "variable-declaration": "nospace"  
}  
],  
"unified-signatures": true,  
"variable-name": false,  
"whitespace": [  
  true,  
  "check-branch",  
  "check-decl",  
  "check-operator",  
  "check-separator",  
  "check-type"  
],  
"directive-selector": [  
  true,  
  "attribute",  
  "app",  
  "camelCase"  
],  
"component-selector": [  
  true,  
  "element",  
  "app",  
  "kebab-case"  
],  
"no-output-on-prefix": true,  
"use-input-property-decorator": true,  
"use-output-property-decorator": true,  
"use-host-property-decorator": true,  
"no-input-rename": true,  
"no-output-rename": true,  
"use-life-cycle-interface": true,  
"use-pipe-transform-interface": true,  
"component-class-suffix": true,  
"directive-class-suffix": true  
}  
}
```

4) protractor.conf.js

- This file contains configuration settings for “protractor” tool, which is used to perform unit testing of components.

- The “protractor” tool is used to execute the test cases that are defined using “jasmine”.

```
// Protractor configuration file, see link for more information
// https://github.com/angular/protractor/blob/master/lib/config.ts

const { SpecReporter } = require('jasmine-spec-reporter');

exports.config = {
  allScriptsTimeout: 11000,
  specs: [
    './e2e/**/*e2e-spec.ts'
  ],
  capabilities: {
    'browserName': 'chrome'
  },
  directConnect: true,
  baseUrl: 'http://localhost:4200/',
  framework: 'jasmine',
  jasmineNodeOpts: {
    showColors: true,
    defaultTimeoutInterval: 30000,
    print: function() {}
  },
  onPrepare() {
    require('ts-node').register({
      project: 'e2e/tsconfig.e2e.json'
    });
    jasmine.getEnv().addReporter(new SpecReporter({ spec: { displayStacktrace: true } }));
  }
};
```

5) karma.conf.js

- This file contains configuration settings for “karma” tool, which is used to execute unit test cases on multiple browsers.

```
// Karma configuration file, see link for more information
// https://karma-runner.github.io/1.0/config/configuration-file.html
```

```
module.exports = function (config) {
  config.set({
    basePath: '',
    frameworks: ['jasmine', '@angular/cli'],
    plugins: [
      require('karma-jasmine'),
      require('karma-chrome-launcher'),
```

```

require('karma-jasmine-html-reporter'),
require('karma-coverage-istanbul-reporter'),
require('@angular/cli/plugins/karma')
],
client:{
  clearContext: false // leave Jasmine Spec Runner output visible in browser
},
coverageIstanbulReporter: {
  reports: [ 'html', 'lcovonly' ],
  fixWebpackSourcePaths: true
},
angularCli: {
  environment: 'dev'
},
reporters: ['progress', 'kjhtml'],
port: 9876,
colors: true,
logLevel: config.LOG_INFO,
autoWatch: true,
browsers: ['Chrome'],
singleRun: false
});
}

```

6) .angular-cli.json

- This file contains configuration settings for “@angular/cli” tool, which is used create, compile and run the application.
- It contains settings such as home page (index.html), startup file name (main.ts), css file name (styles.css) etc.

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "project": {
    "name": "app1"
  },
  "apps": [
    {
      "root": "src",
      "outDir": "dist",
      "assets": [
        "assets",
        "favicon.ico"
      ],
      "index": "index.html",
      "main": "main.ts",
      "polyfills": "polyfills.ts",
      "styles": [
        "styles.css"
      ],
      "scripts": []
    }
  ]
}
```

```
"test": "test.ts",
"tsconfig": "tsconfig.app.json",
"testTsconfig": "tsconfig.spec.json",
"prefix": "app",
"styles": [
  "styles.css"
],
"scripts": [],
"environmentSource": "environments/environment.ts",
"environments": {
  "dev": "environments/environment.ts",
  "prod": "environments/environment.prod.ts"
}
},
"e2e": {
  "protractor": {
    "config": "./protractor.conf.js"
  }
},
"lint": [
{
  "project": "src/tsconfig.app.json",
  "exclude": "**/node_modules/**"
},
{
  "project": "src/tsconfig.spec.json",
  "exclude": "**/node_modules/**"
},
{
  "project": "e2e/tsconfig.e2e.json",
  "exclude": "**/node_modules/**"
},
{
  "test": {
    "karma": {
      "config": "./karma.conf.js"
    }
  },
  "defaults": {
    "styleExt": "css",
    "component": {}
  }
}
```

7) polyfills.ts

- This file contains configuration settings for importing (loading) polyfills which are needed to run angular applications on old browsers such as Internet Explorer.

```
/*
 * This file includes polyfills needed by Angular and is loaded before the app.
 * You can add your own extra polyfills to this file.
 *
 * This file is divided into 2 sections:
 * 1. Browser polyfills. These are applied before loading ZoneJS and are sorted by browsers.
 * 2. Application imports. Files imported after ZoneJS that should be loaded before your
main
*   file.
*
* The current setup is for so-called "evergreen" browsers; the last versions of browsers that
* automatically update themselves. This includes Safari >= 10, Chrome >= 55 (including
Opera),
* Edge >= 13 on the desktop, and iOS 10 and Chrome on mobile.
*
* Learn more in https://angular.io/docs/ts/latest/guide/browser-support.html
*/
*****  

* BROWSER POLYFILLS
*/  
  
/** IE9, IE10 and IE11 requires all of the following polyfills. **/  
//import 'core-js/es6/symbol';  
//import 'core-js/es6/object';  
//import 'core-js/es6/function';  
//import 'core-js/es6/parse-int';  
//import 'core-js/es6/parse-float';  
//import 'core-js/es6/number';  
//import 'core-js/es6/math';  
//import 'core-js/es6/string';  
//import 'core-js/es6/date';  
//import 'core-js/es6/array';  
//import 'core-js/es6/regexp';  
//import 'core-js/es6/map';  
//import 'core-js/es6/weak-map';  
//import 'core-js/es6/set';  
  
/** IE10 and IE11 requires the following for NgClass support on SVG elements */  
// import 'classlist.js'; // Run `npm install --save classlist.js`.  
  
/** IE10 and IE11 requires the following for the Reflect API. */
```

```
// import 'core-js/es6/reflect';

/** Evergreen browsers require these. */
// Used for reflect-metadata in JIT. If you use AOT (and only Angular decorators), you can
// remove.
import 'core-js/es7/reflect';

/**
 * Required to support Web Animations `@angular/platform-browser/animations`.
 * Needed for: All but Chrome, Firefox and Opera. http://caniuse.com/#feat=web-animation
 */
// import 'web-animations-js'; // Run `npm install --save web-animations-js`.

/**
 * By default, zone.js will patch all possible macroTask and DomEvents
 * user can disable parts of macroTask/DomEvents patch by setting following flags
 */
// (window as any).__Zone_disable_requestAnimationFrame = true; // disable patch
requestAnimationFrame
// (window as any).__Zone_disable_on_property = true; // disable patch onProperty such as
onclick
// (window as any).__zone_symbol__BLACK_LISTED_EVENTS = ['scroll', 'mousemove']; // disable patch specified eventNames

/*
 * in IE/Edge developer tools, the addEventListener will also be wrapped by zone.js
 * with the following flag, it will bypass `zone.js` patch for IE/Edge
 */
// (window as any).__Zone_enable_cross_context_check = true;

/**************************************************************************************************
 * Zone JS is required by default for Angular itself.
 */
import 'zone.js/dist/zone'; // Included with Angular CLI.

/**************************************************************************************************
 * APPLICATION IMPORTS
 */

```

8) src/styles.css

- This file contains CSS styles that are applicable for entire application.

```
/* You can add global styles to this file, and also import other style files */
```

9) src/index.html

- This file is the home page (startup page) for the entire application.
- The content of the entire application appears in the same html file only.
- This file invokes the “AppComponent” using <app-root></app-root> tag.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>App1</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></app-root>
</body>
</html>
```

10) src/main.ts

- This is the first typescript file that executes in the angular application.
- It enables the “Production mode” and specifies startup module:

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.log(err));
```

Application Modes in Angular**1. Development Mode:**

- Change Detection occurs twice. Raises error if any difference detected between first attempt and second attempt. This is to identify whether any side effects in change detection.

- It is the default mode.

2. Production Mode

- Change Detection occurs only once..
- **Syntax:** enableProdMode();

Startup Module

- Angular application can has any no. of modules.
- The “startup module” is a module, which needs to be executed first in the angular application.
- By default startup module name is “AppModule”.
- Loading the startup module is also called as “Bootstrapping the module”.
- **Syntax:** platformBrowserDynamic().bootstrapModule(Modulename);

11) src/app/app.module.ts

- This file contains definition of “AppModule”.
- Angular application can has any no. of modules. It should contain atleast one module, that is called as “AppModule”.
- This file imports “AppComponent” from “app.component.ts” file and bootstraps the same in “AppModule”.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

12) src/app/app.component.ts

- This file contains definition of “AppComponent”.
- Angular application can has any no. of components. It should contain atleast one component, that is called as “AppComponent”.

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

@Component({
```

```
    selector: 'app-root',
    templateUrl: './app.component.html',
    styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app';
}
```

- This file specifies path of template file “app.component.html” and css file “app.component.css” file.

13) src/app/app.component.html

- This file contains actual content (html code) of the component.
- Every component should have a template.
- This template content will be rendered into <app-root></app-root> tag at index.html.

```
<div>
  Hello World
</div>
```

14) src/app/app.component.css

- This file contains css styles of “AppComponent”.
- One component can have only one css file.
- By default, this file is empty.

15) src/app/app.component.spec.ts

- This file contains test cases for “AppCompoent”.
- The test case files should have “spec.ts” file extension.

```
import { TestBed, async } from '@angular/core/testing';
import { AppComponent } from './app.component';
describe('AppComponent', () => {
  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [
        AppComponent
      ],
    }).compileComponents();
  }));
  it('should create the app', async(() => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.debugElement.componentInstance;
    expect(app).toBeTruthy();
  }));
  it(`should have as title 'app'`, async(() => {
```

```
const fixture = TestBed.createComponent(AppComponent);
const app = fixture.debugElement.componentInstance;
expect(app.title).toEqual('app');
});
it('should render title in a h1 tag', async(() => {
  const fixture = TestBed.createComponent(AppComponent);
  fixture.detectChanges();
  const compiled = fixture.debugElement.nativeElement;
  expect(compiled.querySelector('h1').textContent).toContain('Welcome to app!');
});
});
```

Components

What is Component?

- The component class represents certain section of the web page. For example, “login form” is represented as a “Login Component”.
- The component class includes “properties” (to store data), “methods” (event handler methods to manipulate data).
- Every “angular 2+ application” contains at-least one component, which is called as “app component”. You can create any no. of components in the project.
- The component is invoked (called) through a custom tag (user-defined tag). For example, “login component” is invoked through <login></login> tag. The custom tag is also called as “selector”.
- The component class should have a decorator called “@Component()”, to define that the class is a component class.

Syntax to create Component:

```
import { Component } from "@angular/core";
@Component( meta data )
class classname
{
  property: datatype = value;

  method( arguments ) : returntype
  {
  }
}
```

Meta Data Properties of Component:

- 1 selector** Represents the selector (tag) to invoke the component.
- 2 template** Represents the template content of the component.
- 3 templateUrl** Represents the html file that has to be rendered when the component is invoked.
- 4 styleUrls** Represents the list of style sheets (css files) that have to be loaded for the component.
- 5 providers** Represents the list of services to be imported into the component.
- 6 animations** Represents the list of animations to be performed in the component.

Modules

What is Module?

- Module is a part of the project.
- Module is a collection of components, directives and pipes that are related to one specific task of the project.
- **Ex:** “Net banking” project contains modules like “Savings account module”, “Credit cards module” etc.
- Every angular application should contain at least one module, which is called as “root module” or “app module”. The “app component” will be a part of the “app module”. Modules can share its components and pipes to other modules.
- Module is a class, with “@NgModule” decorator.

Syntax to create Module:

```
import { NgModule } from "@angular/core";
@NgModule( meta data )
class classname
{
}
```

Meta Data Properties of Module:

1 declarations

Represents the list of components and pipes that are members of the current module.

2 imports

Represents the list of modules that you want to import into the current module.

You must import “BrowserModule” into the browser, which can be imported from “@angular/platform-browser”.

The “BrowserModule” imports “ApplicationModule” from “@angular/core”, “CommonModule” from “@angular/common” and re-exports them.

The “BrowserModule” must be imported only in the “app module (root module)”; we need not import it in other child modules.

Other modules to import: FormsModule, ReactiveFormsModule, BrowserAnimationsModule, HttpClientModule, RouterModule etc.

3 exports

Represents the list of components or pipes that are to be exported to other modules.

4 bootstrap

Represents the component that is to be displayed in the web page. Only “app module” has to bootstrap “app component”. Other modules should not bootstrap any component.

5 providers

Represents list of services to be imported into the module.

Syntax to bootstrap (start) the module into the browser:

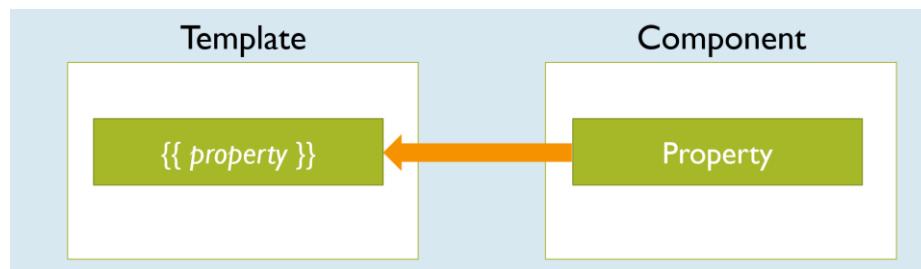
```
import { platformBrowserDynamic } from "@angular/platform-browser-dynamic";
platformBrowserDynamic().bootstrapModule( moduleclassname );
```

Data Bindings

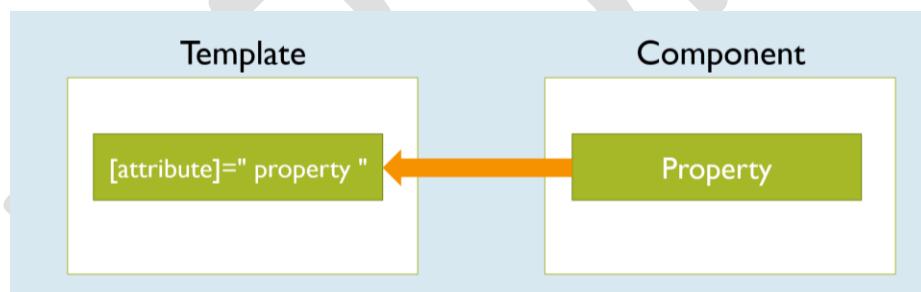
- The “data binding” is the relation between “component” and the “template”.
- When the value of “component” is changed, the “template” will be changed automatically. When the value of “template” is changed, the “component” will be changed automatically.
- Data binding is four types:
 - A) Interpolation Binding
 - B) Property Binding
 - C) Event Binding
 - D) Two-Way Binding

A) Interpolation Binding

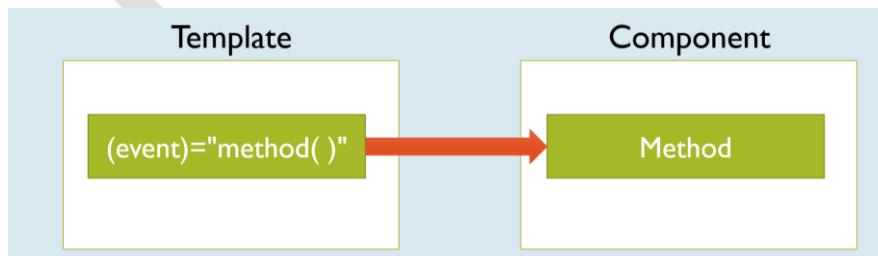
- **Syntax:** {{property}}
- It displays the value of the property in the template.
- When the value of the property is changed, the same value will be automatically updated in the template.

**B) Property Binding**

- **Syntax:** <tag [attribute]=" property "> </tag>
- “Property binding” is used to send data from component to template and assign the same into an attribute of the tag.
- When the value of the property is changed, the same value will be automatically updated in the template.

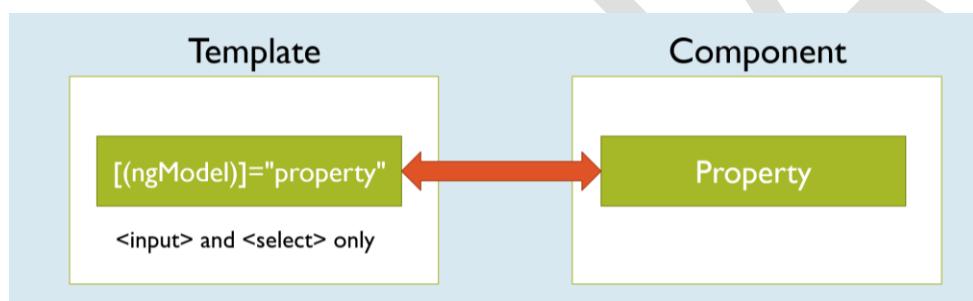
**C) Event Binding**

- **Syntax:** <tag (event)="method()"> </tag>
- It is used to pass event notifications from template to component.



D) Two-Way Binding

- **Syntax:** <tag [(ngModel)]="property"> </tag>
- “Two Way Binding” (a.k.a Two-Way Data Binding) is a combination of both “property binding” and “event binding”.
- When you change the value of “property”, the same will be automatically updated in the “html element”.
- When you change the value of “html element”, the same will be automatically updated in the “property”.
- The “ngModel” is a pre-defined directive, which is used to create two-way binding.
- Two-Way Binding is applicable only for <input> and <select> tags.
- “FormsModule” must be imported in order to use two-way binding.



Data Bindings - Example

Creating Application

- Open Command Prompt and enter the following commands:
- ```
cd c:\angular
ng new app1
```

c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
```

```

"@angular/animations": "^5.2.0",
"@angular/common": "^5.2.0",
"@angular/compiler": "^5.2.0",
"@angular/core": "^5.2.0",
"@angular/forms": "^5.2.0",
"@angular/http": "^5.2.0",
"@angular/platform-browser": "^5.2.0",
"@angular/platform-browser-dynamic": "^5.2.0",
"@angular/router": "^5.2.0",
"core-js": "^2.4.1",
"rxjs": "^5.5.6",
"zone.js": "^0.8.19"
},
"devDependencies": {
"@angular/cli": "~1.7.4",
"@angular/compiler-cli": "^5.2.0",
"@angular/language-service": "^5.2.0",
"@types/jasmine": "~2.8.3",
"@types/jasminewd2": "~2.0.2",
"@types/node": "~6.0.60",
"codemlyzer": "^4.0.1",
"jasmine-core": "~2.8.0",
"jasmine-spec-reporter": "~4.2.1",
"karma": "~2.0.0",
"karma-chrome-launcher": "~2.2.0",
"karma-coverage-istanbul-reporter": "^1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}

```

c:\angular\app1\src\app\app.module.ts

```

import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';

@NgModule({
 declarations: [
 AppComponent
],
 imports: [

```

```
 BrowserModule, FormsModule
],
providers: [],
bootstrap: [AppComponent]
}
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

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

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{
 firstname: string = "Adam";
 lastname: string = "Smith";
 age: number = 20;
 receivenewsletters: boolean = true;
 gender: string = "Male";
 country: string = "India";
 address: string = "http://www.facebook.com";

 ChangeData()
{
 this.firstname = "John";
 this.lastname = "Resig";
 this.age = 30;
 this.receivenewsletters = false;
 this.gender = "Female";
 this.country = "USA";
 }
}
```

c:\angular\app1\src\app\app.component.html

```
<div>
 <h4>Data Bindings</h4>
 Firstname: {{firstname}}

 Lastname: {{lastname}}

 Age: {{age}}

 Receive News Letters: {{receivenewsletters}}

 Gender: {{gender}}

 Country: {{country}}<hr>
```

```
<a [href]="address">Click here<hr>

Firstname: <input type="text" [(ngModel)]="firstname">

Lastname: <input type="text" [(ngModel)]="lastname">

Age: <input type="text" [(ngModel)]="age">

Receive News Letters: <input type="checkbox" [(ngModel)]="receivenewsletters">

Gender:
<input type="radio" [(ngModel)]="gender" value="Male">Male
<input type="radio" [(ngModel)]="gender" value="Female">Female

Country:
<select [(ngModel)]="country">
 <option>India</option>
 <option>UK</option>
 <option>USA</option>
</select>

<input type="button" value="Change Data" (click)="ChangeData()">
</div>
```

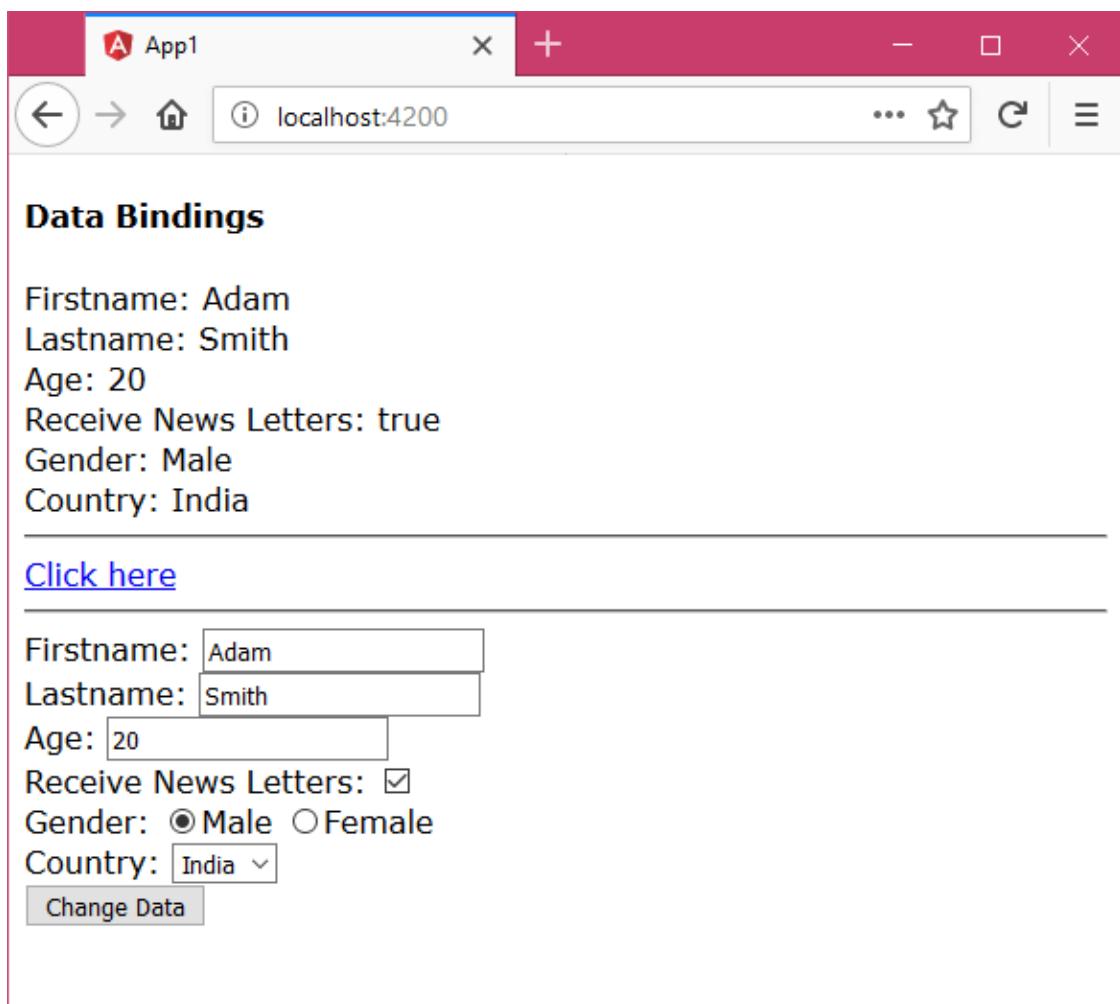
#### Executing the application:

- Open Command Prompt and enter the following commands:

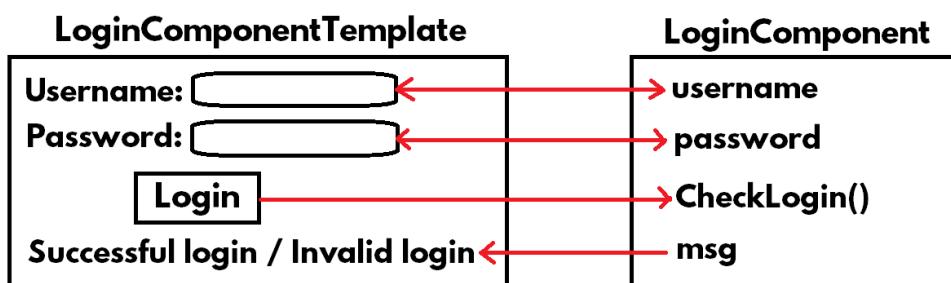
```
cd c:\angular\app1
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



### Login - Example



### Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
```

```
ng new app1
```

**c:\angular\app1\package.json**

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
 },
 "devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
 }
}
```

```
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";

@NgModule({
 declarations: [
 AppComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{
 username: string = "";
 password: string = "";
 msg: string = "";

 CheckLogin(txt1)
 {
 if (this.username == "admin" && this.password == "manager")
 {
 this.msg = "Successful login";
 }
 else
 {
 this.msg = "Invalid login";
 txt1.focus();
 }
 }
}
```

```

 }
}
}

```

c:\angular\app1\src\app\app.component.html

```

<div>
 <form>
 <h4>Login</h4>
 Username: <input type="text" [(ngModel)]="username" name="username" #t1>

 Password: <input type="password" [(ngModel)]="password" name="password">

 <input type="submit" value="Login" (click)="CheckLogin(t1)">

 {{msg}}
 </form>
</div>

```

### Executing the application:

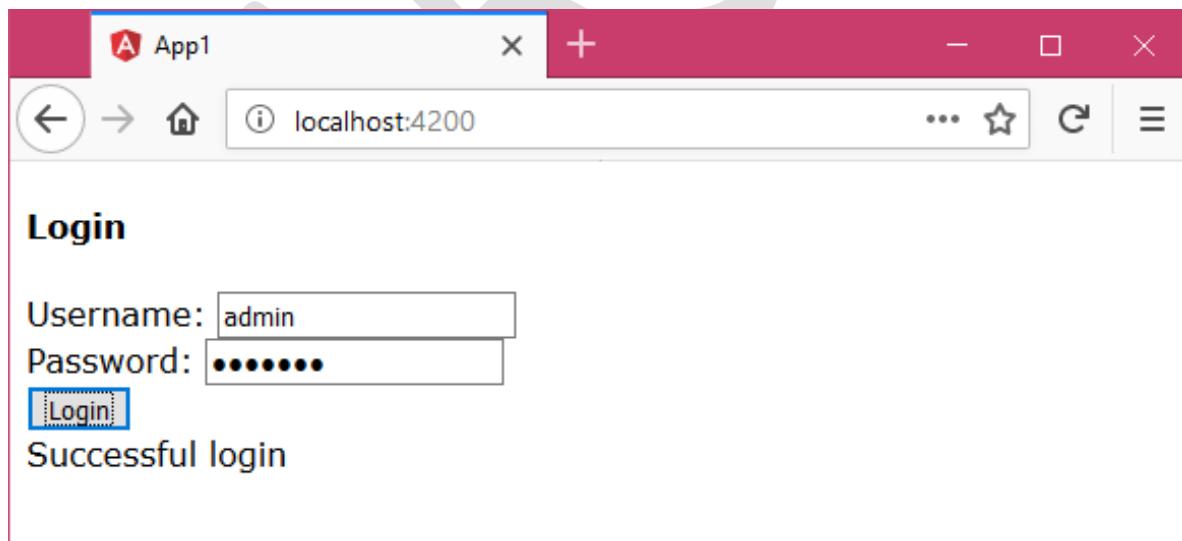
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

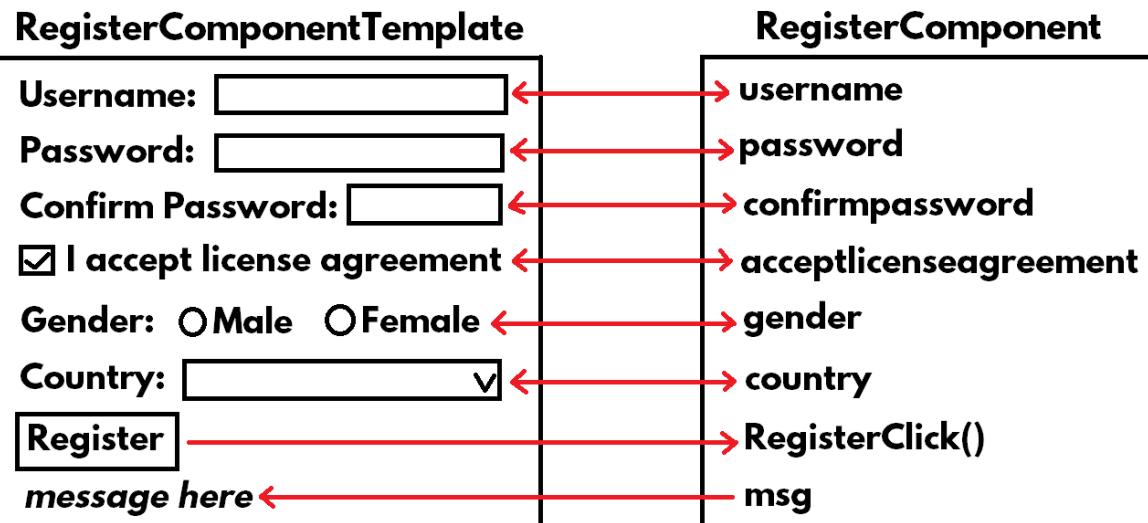
- Open the browser and enter the following URL:

```
http://localhost:4200
```



### Registration Form - Example

- We are going to create a sample registration form, which demonstrates textboxes, checkboxes, radio buttons and dropdownlists.



### Creating Application

- Open Command Prompt and enter the following commands:
- ```
cd c:\angular
ng new app1
```

c:\angular\app1\package.json

```
{
  "name": "app1",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build --prod",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^5.2.0",
    "@angular/common": "^5.2.0",
    "@angular/compiler": "^5.2.0",
    "@angular/core": "^5.2.0",
    "@angular/forms": "^5.2.0",
    "@angular/http": "^5.2.0",
    "@angular/platform-browser": "^5.2.0",
    "@angular/platform-browser-dynamic": "^5.2.0",
    "@angular/router": "^5.2.0",
  }
}
```

```
"core-js": "^2.4.1",
"rxjs": "5.5.6",
"zone.js": "0.8.19"
},
"devDependencies": {
"@angular/cli": "~1.7.4",
"@angular/compiler-cli": "5.2.0",
"@angular/language-service": "5.2.0",
"@types/jasmine": "~2.8.3",
"@types/jasminewd2": "~2.0.2",
"@types/node": "~6.0.60",
"codemlyzer": "4.0.1",
"jasmine-core": "~2.8.0",
"jasmine-spec-reporter": "4.2.1",
"karma": "~2.0.0",
"karma-chrome-launcher": "~2.2.0",
"karma-coverage-istanbul-reporter": "1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule, FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{
  username: string = "";
  password: string = "";
  confirmpassword: string = "";
  acceptlicenseagreement: boolean = false;
  gender: string = "";
  country: string = "";
  msg: string = "";

  RegisterClick()
  {
    this.msg = "Username: " + this.username + "<br>Password: " + this.password +
    "<br>Confirm Password: " + this.confirmpassword + "<br>Accept License Agreement: " +
    this.acceptlicenseagreement + "<br>Gender: " + this.gender + "<br>Country: " + this.country;
  }
}

```

c:\angular\app1\src\app\app.component.html

```

<div>
  <form>
    <h4>Register</h4>
    Username:
    <input type="text" [(ngModel)]="username" name="username"><br>
    Password:
    <input type="password" [(ngModel)]="password" name="password"><br>
    Confirm Password:
    <input type="password" [(ngModel)]="confirmpassword" name="confirmpassword"><br>
    <input type="checkbox" [(ngModel)]="acceptlicenseagreement"
      name="acceptlicenseagreement">
    I accept license agreement<br>
    Gender:
    <input type="radio" [(ngModel)]="gender" value="male" name="gender">Male
    <input type="radio" [(ngModel)]="gender" value="female" name="gender">Female
    <br>
    Country:
    <select [(ngModel)]="country" name="country">
      <option>Please Select</option>

```

```

<option>India</option>
<option>USA</option>
<option>UK</option>
<option>Japan</option>
</select><br>
<input type="submit" value="Register" (click)="RegisterClick()"><br>
<div [innerHTML]="msg"></div>
</form>
</div>

```

Executing the application:

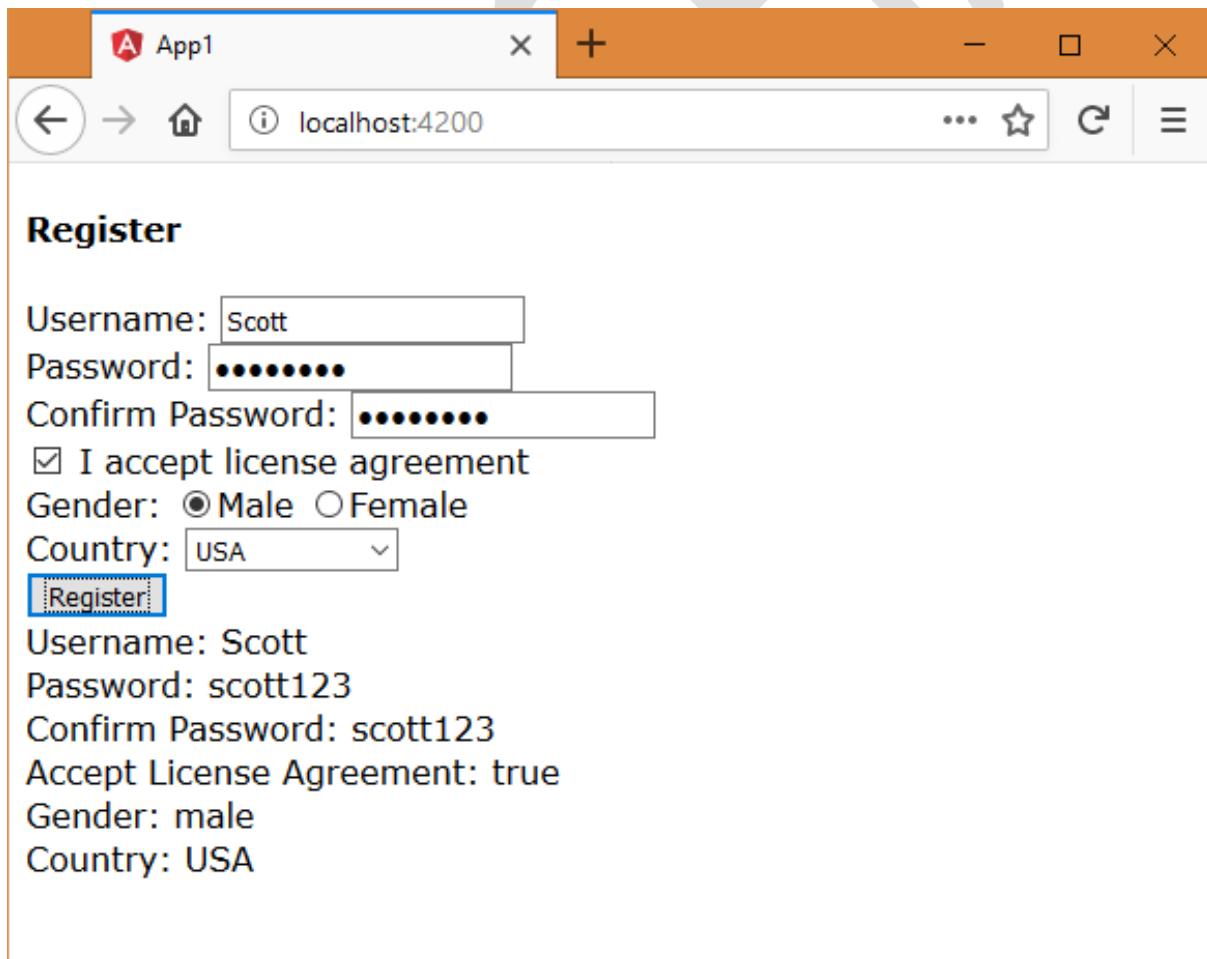
- Open Command Prompt and enter the following commands:

cd c:\angular\app1

ng serve

- Open the browser and enter the following URL:

http://localhost:4200



Built-in Directives

Style

- It is used to set the CSS property value dynamically at run time.
- When the value of component property is changed, the value of css property will be automatically gets changed.

Syntax:

```
<tag [style.cssproperty] = "component property">
</tag>
```

Style - Example

Creating Application

- Open Command Prompt and enter the following commands:
- ```
cd c:\angular
ng new app1
```

c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
 },
 "devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0"
 }
}
```

```
"@angular/compiler-cli": "^5.2.0",
"@angular/language-service": "^5.2.0",
"@types/jasmine": "~2.8.3",
"@types/jasminewd2": "~2.0.2",
"@types/node": "~6.0.60",
"codemirror": "^4.0.1",
"jasmine-core": "~2.8.0",
"jasmine-spec-reporter": "~4.2.1",
"karma": "~2.0.0",
"karma-chrome-launcher": "~2.2.0",
"karma-coverage-istanbul-reporter": "^1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";

@NgModule({
 declarations: [
 AppComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent {
 marks: number = 70;
```

```

mycolor: string = "";

constructor()
{
 if (this.marks >= 35)
 {
 this.mycolor = "green";
 }
 else
 {
 this.mycolor = "red";
 }
}

```

c:\angular\app1\src\app\app.component.html

```

<div>
 <h4>Style</h4>
 <div [style.color]="mycolor">{{marks}}</div>
</div>

```

### Executing the application:

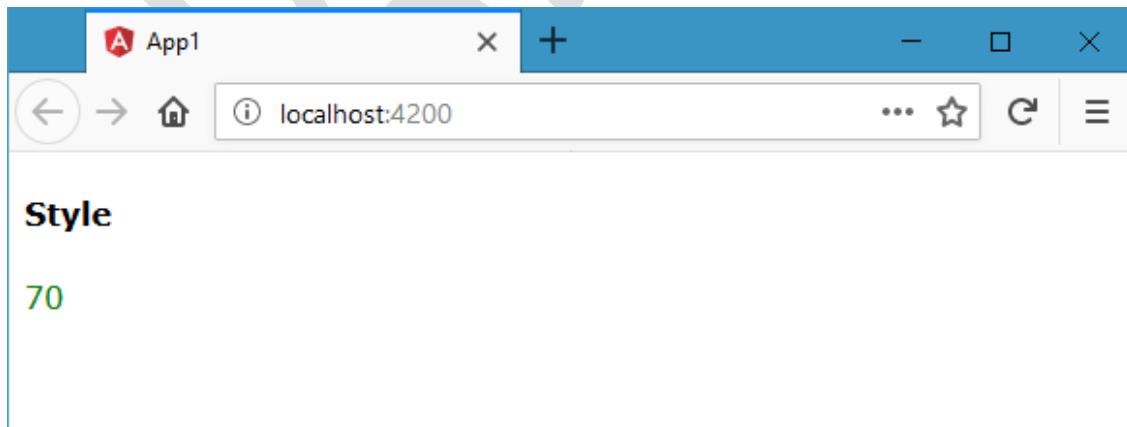
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



### ngClass

- It is used to set the css class name dynamically at run time.
- When the value of component property is changed, the css class will be automatically changed.
- Use this directive to set styles with multiple properties, conditionally at runtime.

### Syntax:

```
<tag [ngClass] = "component property">
</tag>
```

### ngClass - Example

#### Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
```

```
ng new app1
```

c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
},
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",

```

```
"karma-chrome-launcher": "~2.2.0",
"karma-coverage-istanbul-reporter": "^1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

---

**c:\angular\app1\src\app\app.module.ts**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';

@NgModule({
 declarations: [
 AppComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

---

**c:\angular\app1\src\app\app.component.ts**

```
import { Component } from "@angular/core";

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{
 marks: number = 70;
 myclass: string = "";

 constructor()
 {
 if (this.marks >= 35)
 {
 this.myclass = "class1";
 }
 else
```

```
{
 this.myclass = "class2";
}
}
```

c:\angular\app1\src\app\app.component.html

```
<div>
<h4>ngClass</h4>
<div [ngClass]="myclass">{{marks}}</div>
</div>
```

c:\angular\app1\src\app\app.component.css

```
.class1
{
 color: green;
 font-size: 30px;
}

.class2
{
 color: red;
 font-size: 26px;
}
```

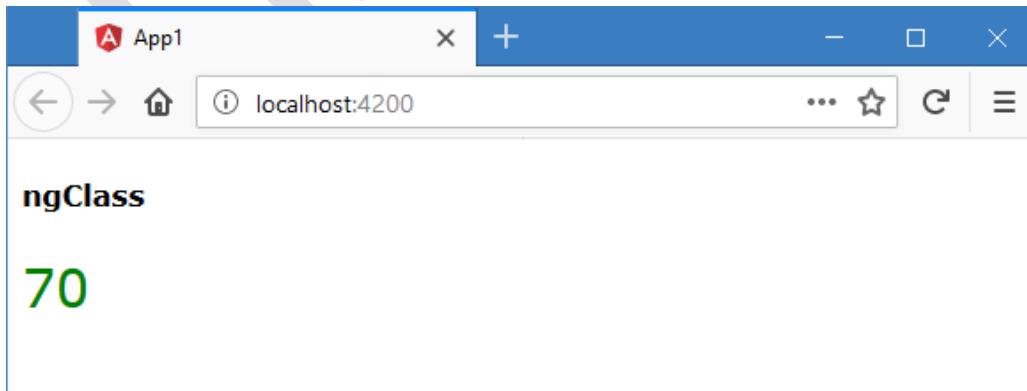
#### Executing the application:

- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



#### ngIf

- The “ngIf” displays the element if the condition is “true”; otherwise the element will be deleted from DOM.

**Syntax:**

```
<tag *ngIf="condition">
</tag>
```

- The “ngIf” must be prefixed with “\*”, to mark that it accepts “micro syntax”, which is not just an “expression”, it accepts its own syntax.
- Use “ngIf” if you want to display some content based on the condition. The content appears when the condition is true, it disappears when the condition is false.

**ngIf - Example****Creating Application**

- Open Command Prompt and enter the following commands:  
cd c:\angular  
ng new app1

c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
},
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",

```

```
"@types/jasmine": "~2.8.3",
"@types/jasminewd2": "~2.0.2",
"@types/node": "~6.0.60",
"codelyzer": "^4.0.1",
"jasmine-core": "~2.8.0",
"jasmine-spec-reporter": "~4.2.1",
"karma": "~2.0.0",
"karma-chrome-launcher": "~2.2.0",
"karma-coverage-istanbul-reporter": "^1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";

@NgModule({
 declarations: [
 AppComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{
 marks: number = 70;
 b: boolean;
```

```

constructor()
{
 if (this.marks >= 35)
 {
 this.b = true;
 }
 else
 {
 this.b = false;
 }
}

```

c:\angular\app1\src\app\app.component.html

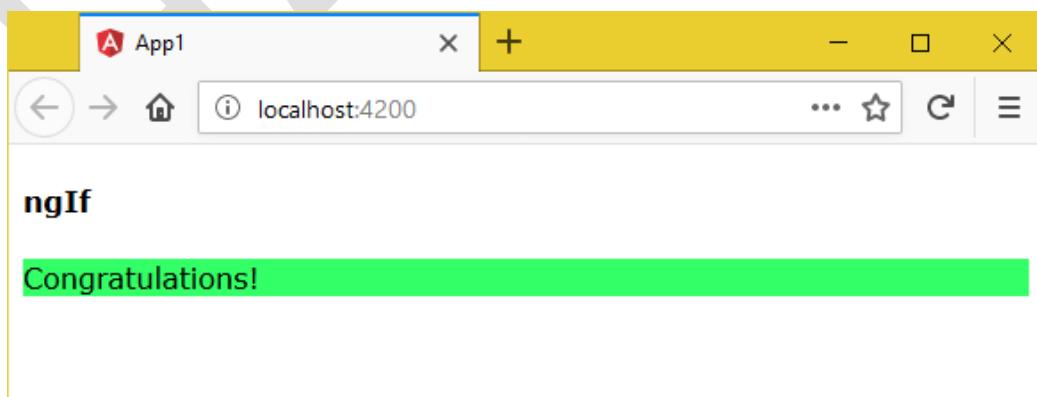
```

<div>
 <h4>ngIf</h4>
 <div *ngIf="b" style="background-color:#33ff66">
 Congratulations!
 </div>
 <div *ngIf="!b" style="background-color:#00ccff">
 Better luck next time!!
 </div>
</div>

```

### Executing the application:

- Open Command Prompt and enter the following commands:  
 cd c:\angular\app1  
 ng serve
- Open the browser and enter the following URL:  
 http://localhost:4200



### ngif and else

- The “ngIf and else” displays one element if it is “true”; otherwise it displays another element.

#### Syntax:

```
<tag *ngIf="condition; then template1; else template2">
</tag>
<ng-template #template1>
...
</ng-template>
<ng-template #template2>
...
</ng-template>
```

- The “ng-template” is a container, inside which you can place any no. of tags.
- Use “ngIf and else”, if you want to display one content for the “true” case, another content for the “false” case.

### ngIf and else - Example

#### Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
```

c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
},
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",

```

```
"core-js": "^2.4.1",
"rxjs": "^5.5.6",
"zone.js": "^0.8.19"
},
"devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';

@NgModule({
 declarations: [
 AppComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

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

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
```

```
styleUrls: ['./app.component.css']
})
export class AppComponent
{
 marks: number = 70;
 b: boolean;

 constructor()
 {
 if (this.marks >= 35)
 {
 this.b = true;
 }
 else
 {
 this.b = false;
 }
 }
}
```

c:\angular\app1\src\app\app.component.html

---

```
<div>
<h4>nglf and else</h4>
<div *ngIf="b; then template1; else template2">
</div>

<ng-template #template1>
<div style="background-color:#33ff66">
 Congratulations!
</div>
</ng-template>

<ng-template #template2>
<div style="background-color:#00ccff">
 Better luck next time!!
</div>
</ng-template>
</div>
```

#### Executing the application:

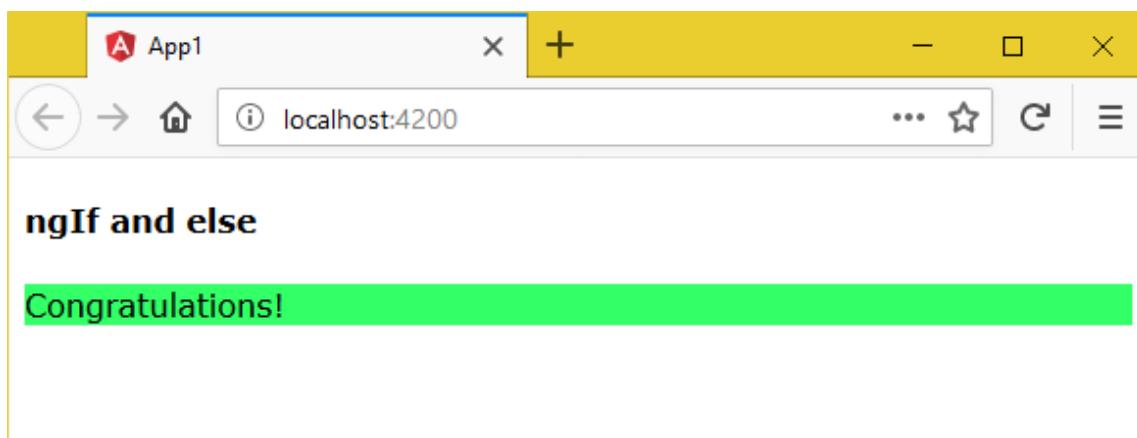
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



## ngSwitch

- The “ngSwitch” checks the value of a variable, whether it matches with any one of the “cases” and displays the element when it matches with anyone.
- Use “ngSwitch” if you want to display some content for every possible value in a variable.

### Syntax:

```
<tag [ngSwitch] = "property">
<tag *ngSwitchCase = "value"></tag>
<tag *ngSwitchCase = "value"></tag>
<tag *ngSwitchCase = "value"></tag>
...
<tag *ngSwitchDefault></tag>
</tag>
```

## ngSwitch - Example

### Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
```

c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
```

```
"lint": "ng lint",
"e2e": "ng e2e"
},
"private": true,
"dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
},
"devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';

@NgModule({
 declarations: [
 AppComponent
],
 imports: [
 BrowserModule, FormsModule
]
})
```

```
],
providers: [],
bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{
 country: string = null;
}
```

c:\angular\app1\src\app\app.component.html

```
<div>
 <h4>ngSwitch</h4>
 <select [(ngModel)]="country">
 <option></option>
 <option>India</option>
 <option>UK</option>
 <option>US</option>
 </select>

 <div [ngSwitch]="country">
 <p *ngSwitchCase="India">
 India details here
 </p>
 <p *ngSwitchCase="UK">
 UK details here
 </p>
 <p *ngSwitchCase="US">
 US details here
 </p>
 <p *ngSwitchDefault>
 Please select any country
 </p>
 </div>
</div>
```

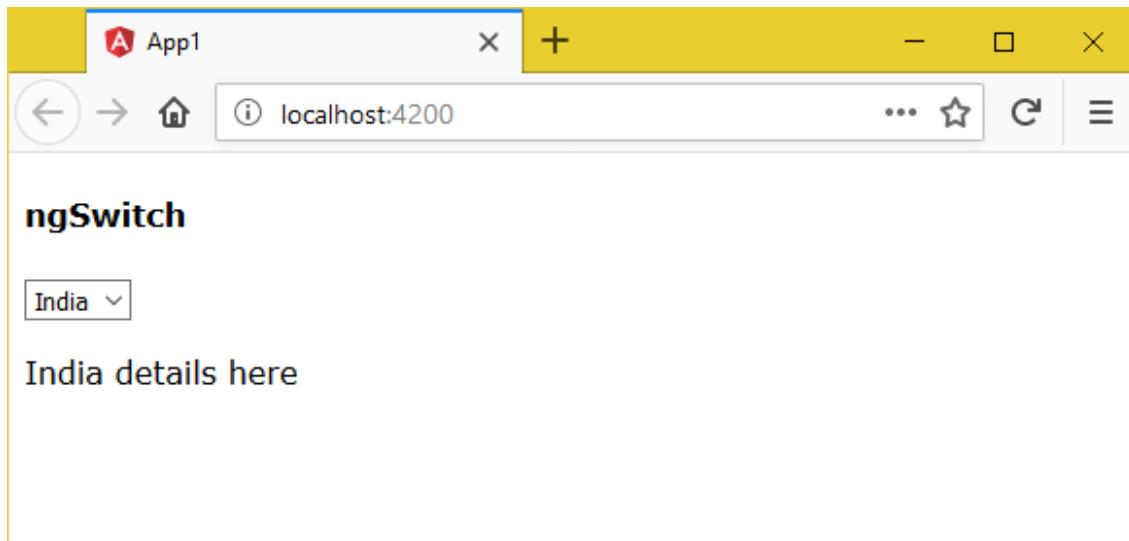
#### Executing the application:

- Open Command Prompt and enter the following commands:  
cd c:\angular\app1

ng serve

- Open the browser and enter the following URL:

http://localhost:4200



## ngFor

### ngFor

- It is used to repeat the tag once for each element in the array. It generates (repeats) the give content once for one element of the array. For example, you are reading friends names from an array and displaying the same as a bulleted list.
- We have to prefix "\*" before "ngFor", as it is the micro syntax. The "micro syntax" means, it's not just a value; we can write some complex code in it.  
arrayname = [ value1, value2, ... ];
- **Ex:** Displaying products list in the online shopping website.
- Use "ngFor" to display list of records. Ex: List of banks, list of salesman and their sales etc.

#### Syntax:

```
<tag *ngFor="let variable of arrayname">
</tag>
```

### ngFor - Example

#### Creating Application

- Open Command Prompt and enter the following commands:

cd c:\angular

ng new app1

## c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
 },
 "devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
 }
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";

@NgModule({
 declarations: [
 AppComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{
 cities: string[] = ["New Delhi", "New York", "New Jersey", "New Mumbai"];
}
```

c:\angular\app1\src\app\app.component.html

```
<div>
<h4>ngFor</h4>

 <li *ngFor="let city of cities">{{city}}

</div>
```

#### Executing the application:

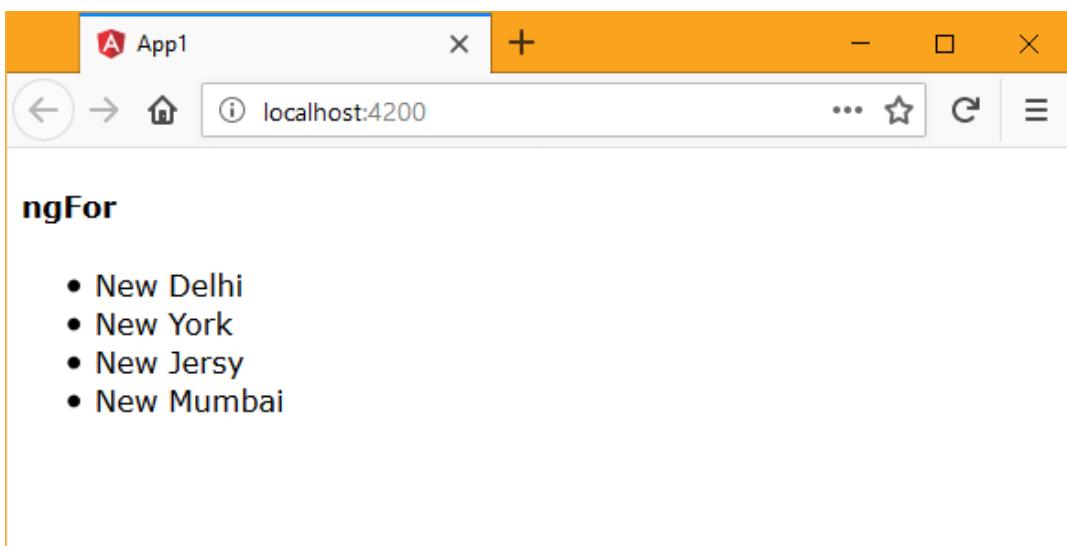
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



### ngFor with Object Array

- First you will store a set of objects inside an array. Read those objects one-by-one by using “ngFor” and display the data in table format.

**Example:** Reading and displaying product names and prices.

- **Create Object Array:**

```
arrayreferencevariable : classname[] =[
 new classname(),
 new classname(),
 ...
];
```

- **ngFor with Object Array:**

```
<tag *ngFor="let variable or arrayreferencevariable">
 variable.property1
 variable.property2
 ...
</tag>
```

### ngFor with Object Array - Example

#### Creating Application

- Open Command Prompt and enter the following commands:  
cd c:\angular

```
ng new app1
cd c:\angular\app1
ng g class Employee
```

**c:\angular\app1\package.json**

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
 },
 "devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codemlyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 }
}
```

```
 "typescript": "~2.5.3"
 }
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";

@NgModule({
 declarations: [
 AppComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\employee.ts

```
export class Employee
{
 empid: number;
 empname: string;
 salary: number;

 constructor(a, b, c)
 {
 this.empid = a;
 this.empname = b;
 this.salary = c;
 }
}
```

c:\angular\app1\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
{
 employees: Employee[] = [
 new Employee(1, "Scott", 4000),
 new Employee(2, "Allen", 7500),
]
}
```

```

 new Employee(3, "Jones", 9200),
 new Employee(4, "James", 9200),
 new Employee(5, "Smith", 8400)
};

}

```

c:\angular\app1\src\app\app.component.html

```

<div>
 <h4>ngFor with Object Array</h4>
 <table border="1">
 <tr>
 <th>Emp ID</th>
 <th>Emp Name</th>
 <th>Salary</th>
 </tr>
 <tr *ngFor="let employee of employees">
 <td>{{employee.empid}}</td>
 <td>{{employee.empname}}</td>
 <td>{{employee.salary}}</td>
 </tr>
 </table>
</div>

```

#### Executing the application:

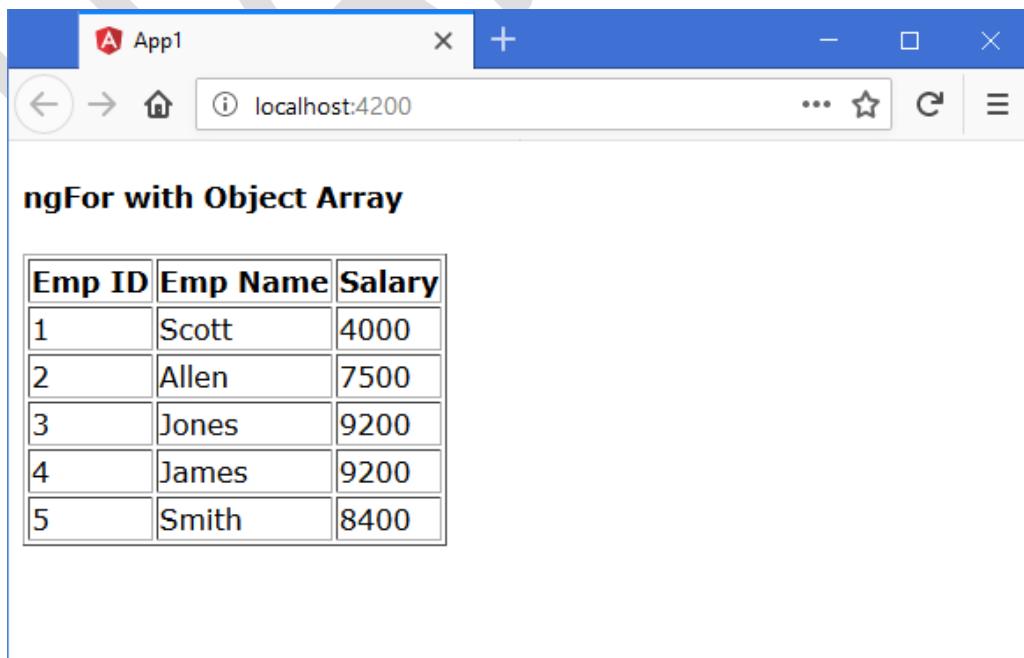
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



## ngFor with Add, Remove

- We can allow the user to add new records (objects) to existing array. The user can also delete existing records.
- We use “push” function to add new object to array.
- We use “splice” function to remove existing object from the array.

### Steps:

- **Adding element to array:**

```
arrayvariable.push(value);
```

- **Removing element from array:**

```
arrayvariable.splice(index, count);
```

## ngFor with Add, Remove - Example

### Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g class Employee
```

c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
```

```
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
 },
 "devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
 }
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";

@NgModule({
 declarations: [
 AppComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\employee.ts

```
export class Employee {
 empid: number;
```

```
empname: string;
salary: number;

constructor(a, b, c)
{
 this.empid = a;
 this.empname = b;
 this.salary = c;
}
```

c:\angular\app1\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
{
 employees: Employee[] = [
 new Employee(1, "Scott", 4000),
 new Employee(2, "Allen", 7500),
 new Employee(3, "Jones", 9200),
 new Employee(4, "James", 9200),
 new Employee(5, "Smith", 8400)
];

 newemployee: Employee = new Employee(null, null, null);

 onInsertClick()
 {
 this.employees.push(new Employee(this.newemployee.empid, this.newemployee.empname,
 this.newemployee.salary));

 this.newemployee.empid = null;
 this.newemployee.empname = null;
 this.newemployee.salary = null;
 }

 onDeleteClick(n)
 {
 if(confirm("Are you sure to delete this employee?"))
 {
 this.employees.splice(n, 1);
 }
 }
}
```

---

**c:\angular\app1\src\app\app.component.html**

```

<div>
 <h4>ngFor with Add, Remove</h4>
 <table border="1">
 <tr>
 <th>Emp ID</th>
 <th>Emp Name</th>
 <th>Salary</th>
 <th></th>
 </tr>
 <tr *ngFor="let employee of employees; let i = index">
 <td>{{employee.empid}}</td>
 <td>{{employee.empname}}</td>
 <td>{{employee.salary}}</td>
 <td><input type="button" value="Delete" (click)="onDeleteClick(i)"></td>
 </tr>
 <tr>
 <td><input type="text" [(ngModel)]="newemployee.empid" placeholder="Emp ID"></td>
 <td><input type="text" [(ngModel)]="newemployee.empname" placeholder="Emp Name"></td>
 <td><input type="text" [(ngModel)]="newemployee.salary" placeholder="Salary"></td>
 <td><input type="button" value="Insert" (click)="onInsertClick()"></td>
 </tr>
 </table>
</div>

```

**Executing the application:**

- Open Command Prompt and enter the following commands:

cd c:\angular\app1

ng serve

- Open the browser and enter the following URL:

<http://localhost:4200>

| Emp ID | Emp Name | Salary |        |
|--------|----------|--------|--------|
| 1      | Scott    | 4000   | Delete |
| 2      | Allen    | 7500   | Delete |
| 3      | Jones    | 9200   | Delete |
| 4      | James    | 9200   | Delete |
| 5      | Smith    | 8400   | Delete |
| Emp ID | Emp Name | Salary | Insert |

- Try to add, remove employees.

### ngFor with Searching and Sorting

- We can search for some content in the array.
- We can allow the user to sort the data based on a specific property.
- We use “filter” function to search content. The filter function receives a callback function, which gets executed once for each item in the array, in the sequence. If the callback function returns “true”, the item will be kept; if it returns “false”, the item will be skipped; thus it collects all the items that has “true”, forms an array with those elements and returns the new array.
- We use “sort” function to sort data. The sort function receives a callback function, which gets called for each pair of items in the list. We should return an integer (either negative, 0, or positive). If negative number is returned, the item1 comes first. If the “0” is returned, no changes will be done; original order of items will be kept as it is. If positive number is returned, item2 comes first, item1 is next.

#### Steps:

- Searching:**

```
arrayname.filter((item) => { return true or false; });
```

- Sorting:**

```
arrayname.sort(
```

```
 (item1, item2) =>
```

```
{
```

```
if (item1 is less than item2)
 return -1;
else if (item1 > item2)
 return 1;
else
 return 0;
}
```

### ngFor with Searching and Sorting- Example

#### Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g class Employee
```

#### c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 }
}
```

```
"rxjs": "^5.5.6",
"zone.js": "^0.8.19"
},
"devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';

@NgModule({
 declarations: [
 AppComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\employee.ts

```
export class Employee {
 empid: number;
 empname: string;
 salary: number;

 constructor(a, b, c)
```

```
{
 this.empid = a;
 this.empname = b;
 this.salary = c;
}
}
```

c:\angular\app1\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
{
 originaledemployees: Employee[] = [
 new Employee(1, "Scott", 4000),
 new Employee(2, "Allen", 7500),
 new Employee(3, "Jones", 9200),
 new Employee(4, "Ford", 6900),
 new Employee(5, "Mark", 8400)
];

 employees: Employee[] = [];

 constructor()
 {
 this.employees = this.originaledemployees;
 }

 str: string = "";
 sortcolumn = "empid";
 order = 1;

 onSearchClick()
 {
 this.employees = this.originaledemployees.filter((emp) => { return
 emp.empname.toLowerCase().indexOf(this.str.toLowerCase()) >= 0; });
 }

 onSortClick()
 {
 this.employees = this.originaledemployees.sort((emp1, emp2) =>
 {
 var n = 0;
 if (this.sortcolumn == "empid")
 {
 return (emp1[this.sortcolumn] - emp2[this.sortcolumn]) * this.order;
 }
 });
 }
}
```

```
 else if (this.sortcolumn == "empname")
 {
 return (emp1[this.sortcolumn].charCodeAt(0) - emp2[this.sortcolumn].charCodeAt(0)) * this.order;
 }
 else
 {
 return (emp1[this.sortcolumn] - emp2[this.sortcolumn]) * this.order;
 }
});
```

c:\angular\app1\src\app\app.component.html

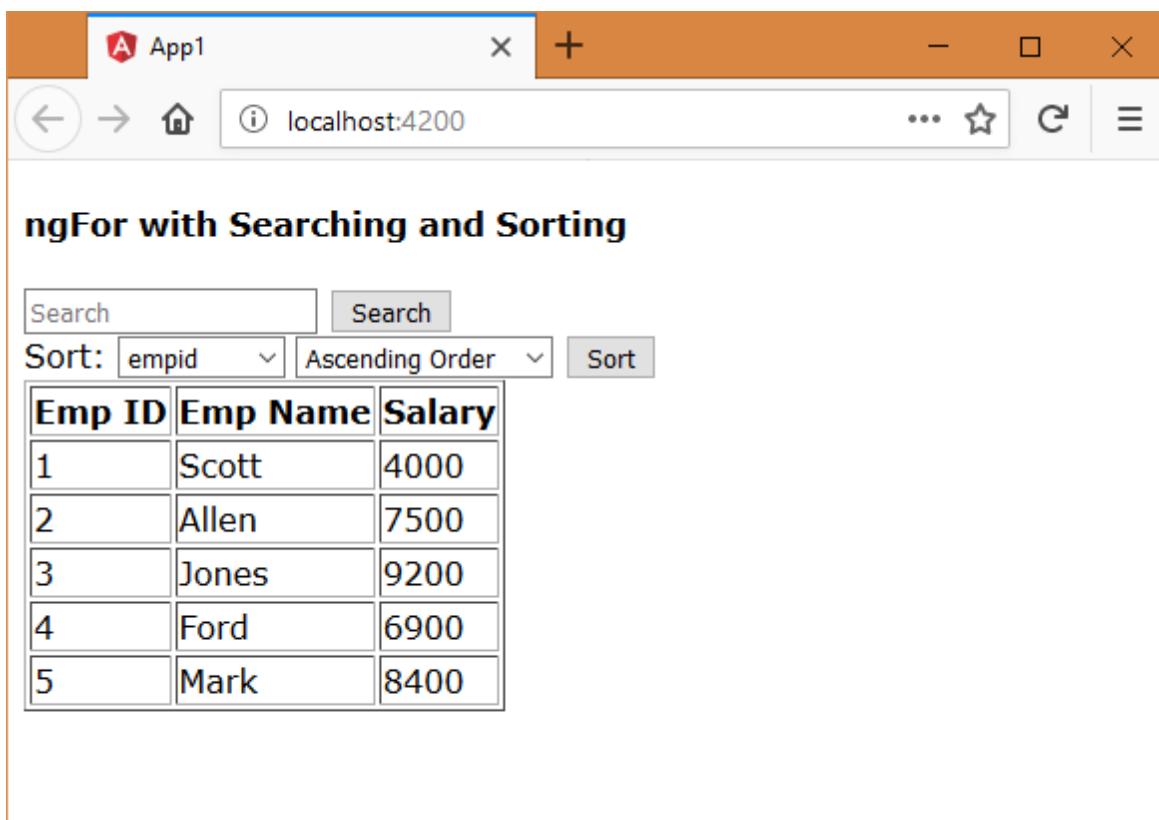
```
<div>
<h4>ngFor with Searching and Sorting</h4>
<input type="text" placeholder="Search" [(ngModel)]="str">
<input type="button" value="Search" (click)="onSearchClick()">

Sort:
<select [(ngModel)]="sortcolumn">
 <option>empid</option>
 <option>empname</option>
 <option>salary</option>
</select>
<select [(ngModel)]="order">
 <option value="1">Ascending Order</option>
 <option value="-1">Descending Order</option>
</select>
<input type="button" value="Sort" (click)="onSortClick()">
<table border="1">
 <tr>
 <th>Emp ID</th>
 <th>Emp Name</th>
 <th>Salary</th>
 </tr>
 <tr *ngFor="let employee of employees; let i = index">
 <td>{{employee.empid}}</td>
 <td>{{employee.empname}}</td>
 <td>{{employee.salary}}</td>
 </tr>
</table>
</div>
```

### **Executing the application:**

- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
ng serve
```
  - Open the browser and enter the following URL:  
<http://localhost:4200>



- Try to click on "Search" and "Sort" buttons.

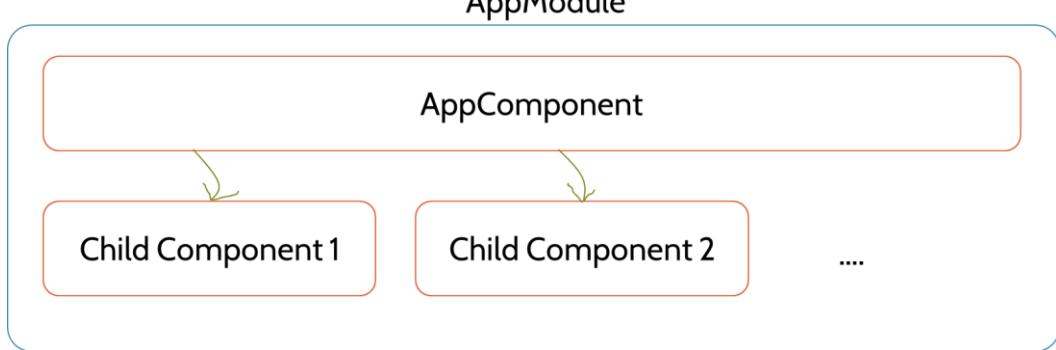
## Multiple Components, Multiple Modules

### Multiple Components

- We can create any no. of components in an application. The component represents a specific section in the web page. The component is a class with properties and methods.
- At root level, only one component should be there, which is called as "AppComponent". All other components should be children of AppComponent.
- We use the child component's selector to invoke the child component in the parent component.

#### Steps:

```
<childselector> </childselector>
```



### Multiple Components - Example

#### Creating Application

- Open Command Prompt and enter the following commands:

```

cd c:\angular
ng new app1
cd c:\angular\app1
ng g component India
ng g component USA
ng g component NewDelhi
ng g component NewMumbai
ng g component NewYork
ng g component Washington

```

#### c:\angular\app1\package.json

```

{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 ...
 }
}

```

```
"@angular/forms": "^5.2.0",
"@angular/http": "^5.2.0",
"@angular/platform-browser": "^5.2.0",
"@angular/platform-browser-dynamic": "^5.2.0",
"@angular/router": "^5.2.0",
"core-js": "^2.4.1",
"rxjs": "^5.5.6",
"zone.js": "^0.8.19"
},
"devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\styles.css

```
.class1
{
 background-color: #91e3e9;
 width: 500px;
 height: 300px;
 margin: 20px;
 float: left;
}

.class2
{
 background-color: #28abdd;
 width: 200px;
 height: 100px;
 margin: 20px;
 float: left;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";
import { IndiaComponent } from './india/india.component';
import { UsaComponent } from './usa/usa.component';
import { NewDelhiComponent } from './new-delhi/new-delhi.component';
import { NewMumbaiComponent } from './new-mumbai/new-mumbai.component';
import { NewYorkComponent } from './new-york/new-york.component';
import { WashingtonComponent } from './washington/washington.component';

@NgModule({
 declarations: [
 AppComponent,
 IndiaComponent,
 UsaComponent,
 NewDelhiComponent,
 NewMumbaiComponent,
 NewYorkComponent,
 WashingtonComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div>
<h4>Multiple Components</h4>
<app-india></app-india>
<app-usa></app-usa>
</div>
```

c:\angular\app1\src\app\india\india.component.ts

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

```
@Component({
 selector: 'app-india',
 templateUrl: './india.component.html',
 styleUrls: ['./india.component.css']
})
export class IndiaComponent implements OnInit {

 constructor() {}

 ngOnInit() {
 }
}
```

c:\angular\app1\src\app\india\india.component.html

```
<div class="class1">
 <h4>India</h4>
 <app-new-delhi></app-new-delhi>
 <app-new-mumbai></app-new-mumbai>
</div>
```

c:\angular\app1\src\app\usa\usa.component.ts

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

@Component({
 selector: 'app-usa',
 templateUrl: './usa.component.html',
 styleUrls: ['./usa.component.css']
})
export class UsaComponent implements OnInit {

 constructor() {}

 ngOnInit() {
 }
}
```

c:\angular\app1\src\app\usa\usa.component.html

```
<div class="class1">
 <h4>United States</h4>
 <app-new-york></app-new-york>
 <app-washington></app-washington>
</div>
```

c:\angular\app1\src\app\new-delhi\new-delhi.component.ts

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

@Component({
 selector: 'app-new-delhi',
```

```
templateUrl: './new-delhi.component.html',
styleUrls: ['./new-delhi.component.css']
})
export class NewDelhiComponent implements OnInit {

constructor() {}

ngOnInit() {
}

}
```

c:\angular\app1\src\app\new-delhi\new-delhi.component.html

```
<div class="class2">
<h4>New Delhi</h4>
</div>
```

c:\angular\app1\src\app\new-mumbai\new-mumbai.component.ts

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

@Component({
selector: 'app-new-mumbai',
templateUrl: './new-mumbai.component.html',
styleUrls: ['./new-mumbai.component.css']
})
export class NewMumbaiComponent implements OnInit {

constructor() {}

ngOnInit() {
}

}
```

c:\angular\app1\src\app\new-mumbai\new-mumbai.component.html

```
<div class="class2">
<h4>New Mumbai</h4>
</div>
```

c:\angular\app1\src\app\new-york\new-york.component.ts

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

@Component({
selector: 'app-new-york',
templateUrl: './new-york.component.html',
styleUrls: ['./new-york.component.css']
})
export class NewYorkComponent implements OnInit {

constructor() {}
```

```
ngOnInit() {
}
}
```

c:\angular\app1\src\app\new-york\new-york.component.html

```
<div class="class2">
 <h4>New York</h4>
</div>
```

c:\angular\app1\src\app\washington\washington.component.ts

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

@Component({
 selector: 'app-washington',
 templateUrl: './washington.component.html',
 styleUrls: ['./washington.component.css']
})
export class WashingtonComponent implements OnInit {

 constructor() {}

 ngOnInit() {}
}
```

c:\angular\app1\src\app\washington\washington.component.html

```
<div class="class2">
 <h4>Washington</h4>
</div>
```

### Executing the application:

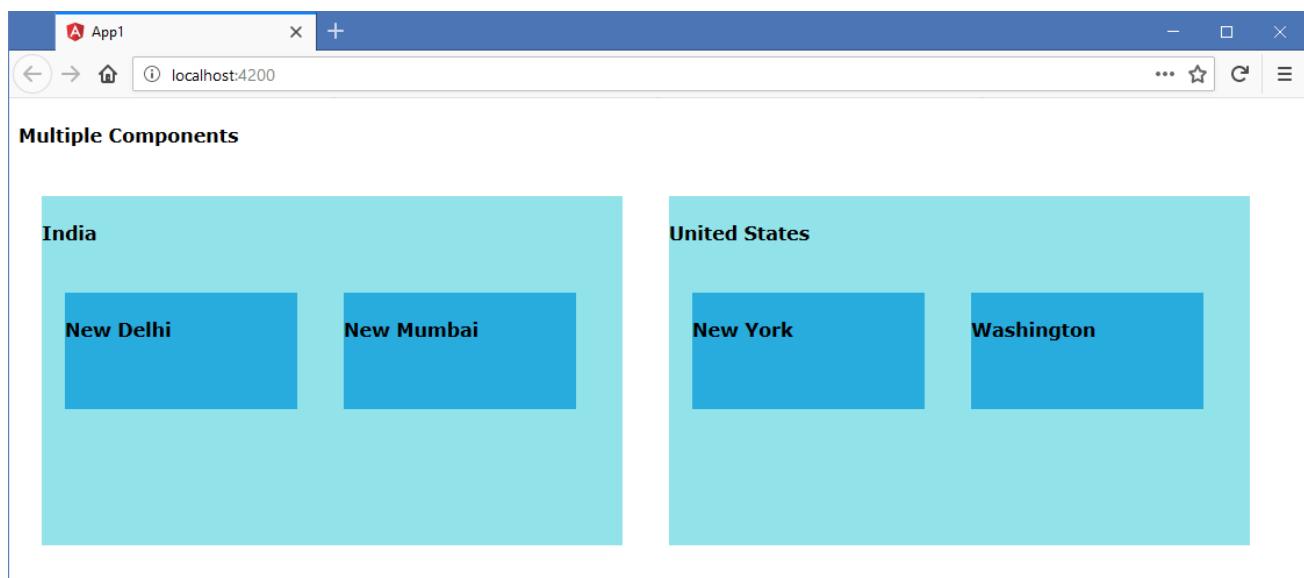
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

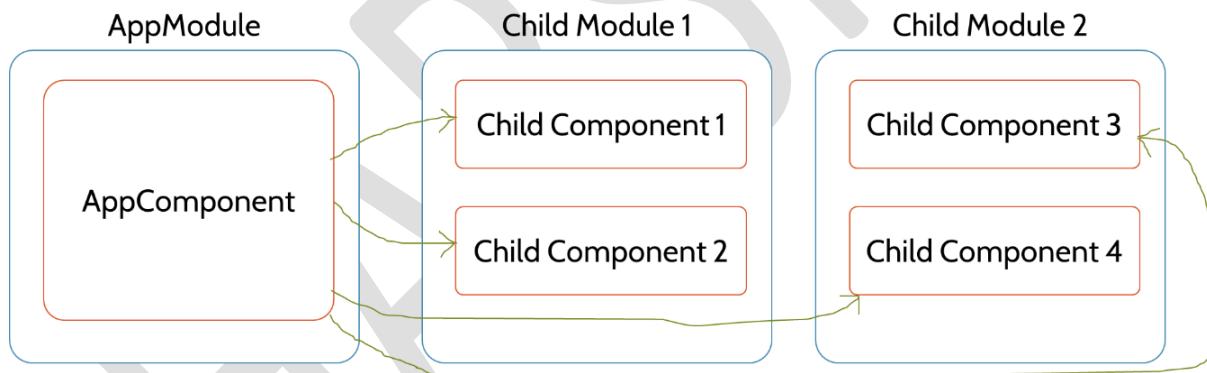
- Open the browser and enter the following URL:

```
http://localhost:4200
```



### Multiple Modules

- The “angular 2+ application” can have any no. of modules.
- “App Module” is the main module; it contains “AppComponent”.
- Child modules contain child components, which can be called in the “AppComponent”.



### Multiple Modules - Example

#### Creating Application

- Open Command Prompt and enter the following commands:

```

cd c:\angular
ng new app1
cd c:\angular\app1
ng g component India
ng g component USA
ng g component NewDelhi
ng g component NewMumbai

```

```
ng g component NewYork
ng g component Washington
ng g module IndiaModule
ng g module UsaModule
```

**c:\angular\app1\package.json**

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
 },
 "devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "~1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 }
}
```

```
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
 }
}
```

c:\angular\app1\src\styles.css

```
.class1
{
 background-color: #91e3e9;
 width: 500px;
 height: 300px;
 margin: 20px;
 float: left;
}

.class2
{
 background-color: #28abdd;
 width: 200px;
 height: 100px;
 margin: 20px;
 float: left;
}
```

c:\angular\app1\src\india-module\india-module.module.ts

```
import { NgModule } from '@angular/core';
import { NewDelhiComponent } from './new-delhi/new-delhi.component';
import { NewMumbaiComponent } from './new-mumbai/new-mumbai.component';

@NgModule({
 declarations: [NewDelhiComponent, NewMumbaiComponent],
 exports: [NewDelhiComponent, NewMumbaiComponent]
})
export class IndiaModuleModule {}
```

c:\angular\app1\src\usa-module\usa-module.module.ts

```
import { NgModule } from '@angular/core';
import { NewYorkComponent } from './new-york/new-york.component';
import { WashingtonComponent } from './washington/washington.component';

@NgModule({
 declarations: [NewYorkComponent, WashingtonComponent],
 exports: [NewYorkComponent, WashingtonComponent]
})
export class UsaModuleModule {}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";
```

```
import { IndiaComponent } from './india/india.component';
import { UsaComponent } from './usa/usa.component';
import { IndiaModuleModule } from "./india-module/india-module.module";
import { UsaModuleModule } from "./usa-module/usa-module.module";

@NgModule({
 declarations: [
 AppComponent,
 IndiaComponent,
 UsaComponent
],
 imports: [
 BrowserModule, FormsModule, IndiaModuleModule, UsaModuleModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div>
 <h4>Multiple Modules</h4>
 <app-india></app-india>
 <app-usa></app-usa>
</div>
```

c:\angular\app1\src\app\india\india.component.ts

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

@Component({
 selector: 'app-india',
 templateUrl: './india.component.html',
 styleUrls: ['./india.component.css']
})
export class IndiaComponent implements OnInit {

 constructor() {}

 ngOnInit() {
```

```
}
```

```
}
```

c:\angular\app1\src\app\india\india.component.html

```
<div class="class1">
<h4>India</h4>
<app-new-delhi></app-new-delhi>
<app-new-mumbai></app-new-mumbai>
</div>
```

c:\angular\app1\src\app\usa\usa.component.ts

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

@Component({
 selector: 'app-usa',
 templateUrl: './usa.component.html',
 styleUrls: ['./usa.component.css']
})
export class UsaComponent implements OnInit {

 constructor() {}

 ngOnInit() {
 }

}
```

c:\angular\app1\src\app\usa\usa.component.html

```
<div class="class1">
<h4>United States</h4>
<app-new-york></app-new-york>
<app-washington></app-washington>
</div>
```

c:\angular\app1\src\app\new-delhi\new-delhi.component.ts

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

@Component({
 selector: 'app-new-delhi',
 templateUrl: './new-delhi.component.html',
 styleUrls: ['./new-delhi.component.css']
})
export class NewDelhiComponent implements OnInit {

 constructor() {}

 ngOnInit()
 {
 }

}
```

```
}
```

c:\angular\app1\src\app\new-delhi\new-delhi.component.html

```
<div class="class2">
<h4>New Delhi</h4>
</div>
```

c:\angular\app1\src\app\new-mumbai\new-mumbai.component.ts

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

@Component({
 selector: 'app-new-mumbai',
 templateUrl: './new-mumbai.component.html',
 styleUrls: ['./new-mumbai.component.css']
})
export class NewMumbaiComponent implements OnInit {

 constructor() {}

 ngOnInit() {
 }

}
```

c:\angular\app1\src\app\new-mumbai\new-mumbai.component.html

```
<div class="class2">
<h4>New Mumbai</h4>
</div>
```

c:\angular\app1\src\app\new-york\new-york.component.ts

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

@Component({
 selector: 'app-new-york',
 templateUrl: './new-york.component.html',
 styleUrls: ['./new-york.component.css']
})
export class NewYorkComponent implements OnInit {

 constructor() {}

 ngOnInit() {
 }

}
```

c:\angular\app1\src\app\new-york\new-york.component.html

```
<div class="class2">
<h4>New York</h4>
</div>
```

c:\angular\app1\src\app\washington\washington.component.ts

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

@Component({
 selector: 'app-washington',
 templateUrl: './washington.component.html',
 styleUrls: ['./washington.component.css']
})
export class WashingtonComponent implements OnInit {

 constructor() {}

 ngOnInit() {
 }
}
```

c:\angular\app1\src\app\washington\washington.component.html

```
<div class="class2">
<h4>Washington</h4>
</div>
```

### Executing the application:

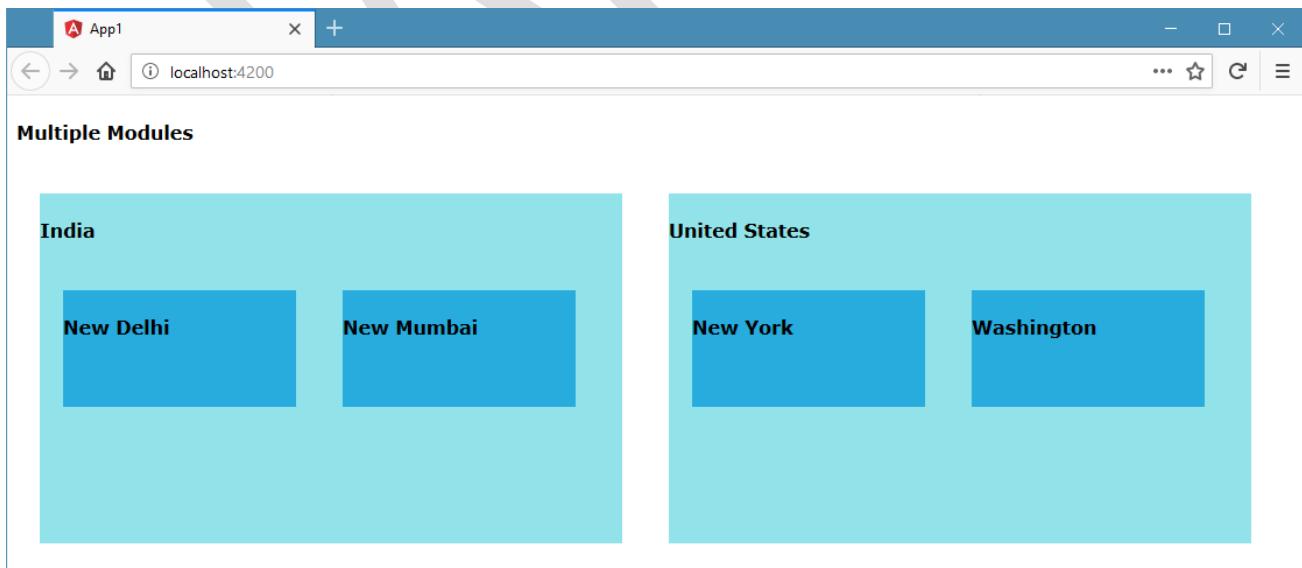
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



## Children of Components

### Sharing Data from Parent Component to Child Component

- We can share the data from parent component to child component using “property binding”.
- Assign the value of “parent component’s property” to “child component’s property”, using “property binding”.
- Set @Input() decorator for the child component’s property to accept value from parent component’s property. You can import “Input” decorator from “@angular/core” package.

#### Steps:

- Import “Input”:

```
import { Input } from "@angular/core";
```

- Create data in parent property at parent component:

```
class parentcomponent
{
 parentproperty: datatype;
 ...
}
```

- Pass data from parent property to child property:

```
<child [childproperty]="parentproperty" ...>
</child>
```

- Receive data into child property at child component:

```
class childcomponent
{
 @Input() childproperty : datatype;
 ...
}
```

### Sharing Data from Parent Component to Child Component - Example

#### Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
```

```
ng new app1
```

```
cd c:\angular\app1
```

```
ng g component company
```

```
ng g component employee
```

### c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
 },
 "devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
 }
}
```

```
}
```

c:\angular\app1\src\styles.css

```
.class1
{
 background-color: #0094ff;
 margin: 10px;
 padding: 10px;
 height: 300px;
}

.class2
{
 background-color: #47f3aa;
 margin: 10px;
 padding: 10px;
 height: 100px;
}
```

c:\angular\app1\src\app\app.module.ts

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

import { AppComponent } from './app.component';
import { CompanyComponent } from './company/company.component';
import { EmployeeComponent } from './employee/employee.component';

@NgModule({
 declarations: [
 AppComponent,
 CompanyComponent,
 EmployeeComponent
],
 imports: [
 BrowserModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

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

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
```

```
)
export class AppComponent {
}
```

c:\angular\app1\src\app\app.component.html

```
<div>
 <h4>Sharing Data from Parent to Child using Property Binding</h4>
 <app-company></app-company>
</div>
```

c:\angular\app1\src\app\company\company.component.ts

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

@Component({
 selector: 'app-company',
 templateUrl: './company.component.html',
 styleUrls: ['./company.component.css']
)
export class CompanyComponent implements OnInit {

 phonenumber: string = "9898923810";
 constructor() {}

 ngOnInit() {
 }

}
```

c:\angular\app1\src\app\company\company.component.html

```
<div class="class1">
 <h4>Company</h4>
 <app-employee [x]="phonenumber"></app-employee>
</div>
```

c:\angular\app1\src\app\employee\employee.component.ts

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

@Component({
 selector: 'app-employee',
 templateUrl: './employee.component.html',
 styleUrls: ['./employee.component.css']
)
export class EmployeeComponent implements OnInit {

 @Input() x: string;

 constructor() {}

 ngOnInit() {
```

```
}
```

```
}
```

c:\angular\app1\src\app\employee\employee.component.html

```
<div class="class2">
<h4>Employee</h4>
<p>Company Phone number: {{x}}</p>
</div>
```

### Executing the application:

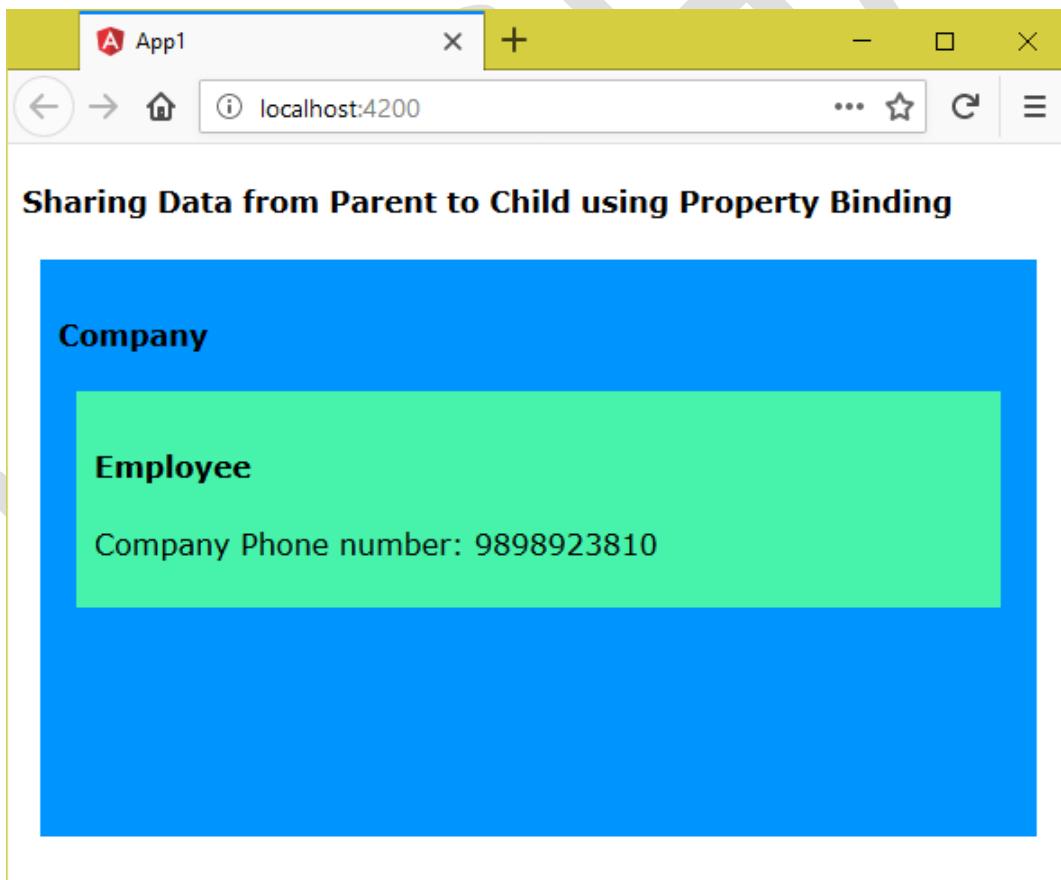
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



### ViewChild

- The “ViewChild” represents an element, which is a child of the view (template) of the component.
- ViewChild is used to access an element, that is present in the view (template) of the component.
- ViewChild can contain a child element of a specific type (class).

- ViewChild is used to access properties / methods of the child.

### Steps:

- Import “ViewChild”:

```
import { ViewChild } from "@angular/core";
```

- Create ViewChild property:

```
class parentcomponent
{
 @ViewChild(classname) propertynname: classname;
 ...
}
```

- Access properties / methods of the child element, using ViewChild’s property:

```
this.propertynname.property
this.propertynname.method()
```

### **ViewChild - Example**

#### **Creating Application**

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g component Company
ng g component Employee
```

#### **c:\angular\app1\package.json**

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
```

```
"@angular/animations": "^5.2.0",
"@angular/common": "^5.2.0",
"@angular/compiler": "^5.2.0",
"@angular/core": "^5.2.0",
"@angular/forms": "^5.2.0",
"@angular/http": "^5.2.0",
"@angular/platform-browser": "^5.2.0",
"@angular/platform-browser-dynamic": "^5.2.0",
"@angular/router": "^5.2.0",
"core-js": "^2.4.1",
"rxjs": "^5.5.6",
"zone.js": "^0.8.19"
},
"devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\styles.css

```
.class1
{
 border: 2px solid red;
 margin: 10px;
 padding: 5px;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from "@angular/forms";

import { AppComponent } from './app.component';
import { CompanyComponent } from './company/company.component';
import { EmployeeComponent } from './employee/employee.component';
```

```
@NgModule({
 declarations: [
 AppComponent,
 CompanyComponent,
 EmployeeComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

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

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div class="class1">
 <h4>App</h4>
 <app-company></app-company>
</div>
```

c:\angular\app1\src\app\company\company.component.ts

```
import { Component, ViewChild } from '@angular/core';
import { EmployeeComponent } from './employee/employee.component';

@Component({
 selector: 'app-company',
 templateUrl: './company.component.html',
 styleUrls: ['./company.component.css']
})
export class CompanyComponent
{
 companyname: string = "ABC Company";

 @ViewChild(EmployeeComponent) emp: EmployeeComponent;

 onClickMeClicked()
```

```
{
 console.log(this.emp);
 this.emp.empname = "John";
}
}
```

c:\angular\app1\src\app\company\company.component.html

```
<div class="class1">
 <h5>Company</h5>
 {{companyname}}

 <input type="button" value="Click me" (click)="onClickMeClicked()">
 <app-employee></app-employee>
</div>
```

c:\angular\app1\src\app\employee\employee.component.ts

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

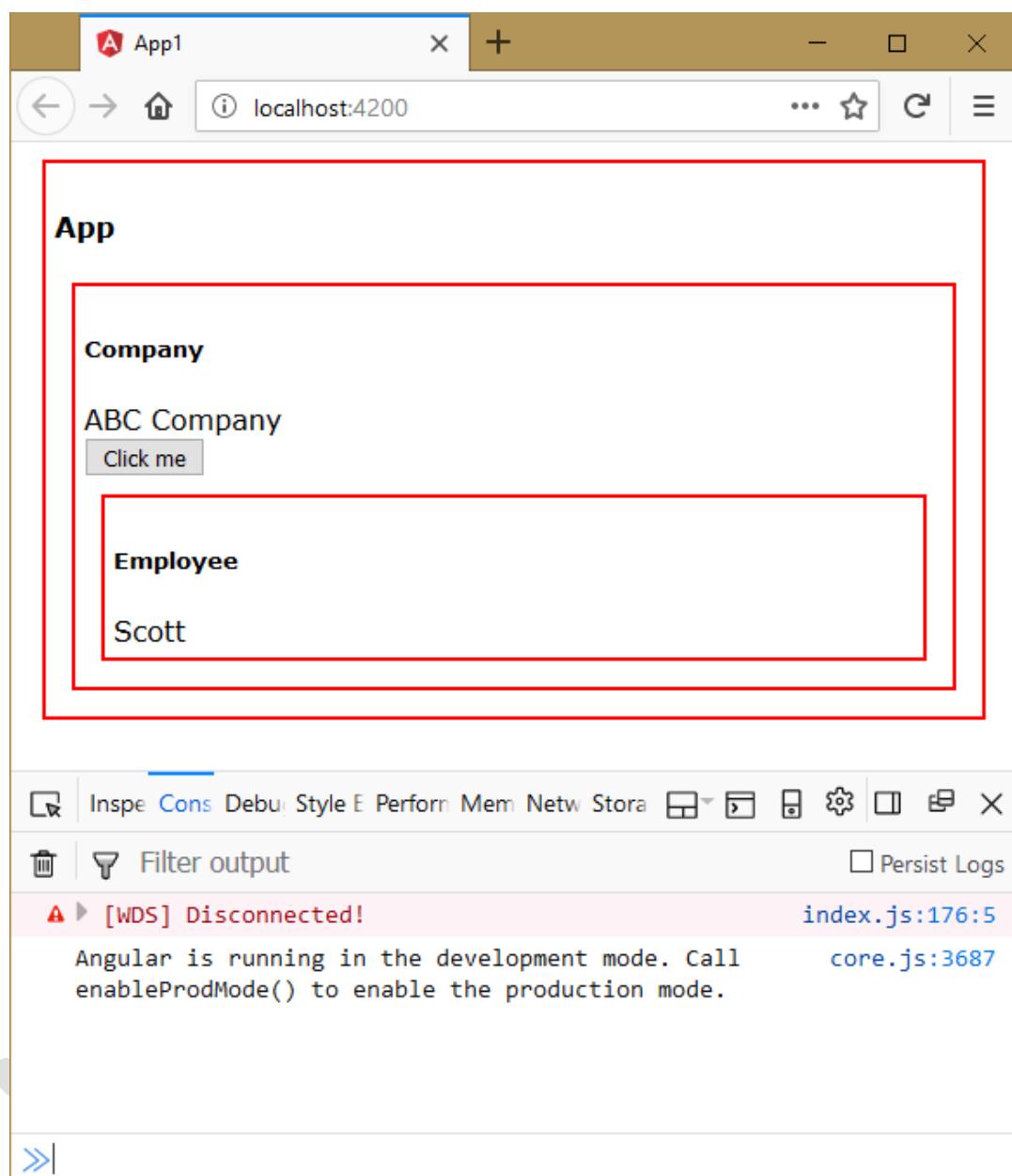
@Component({
 selector: 'app-employee',
 templateUrl: './employee.component.html',
 styleUrls: ['./employee.component.css']
)
export class EmployeeComponent
{
 empname: string = "Scott";
}
```

c:\angular\app1\src\app\employee\employee.component.html

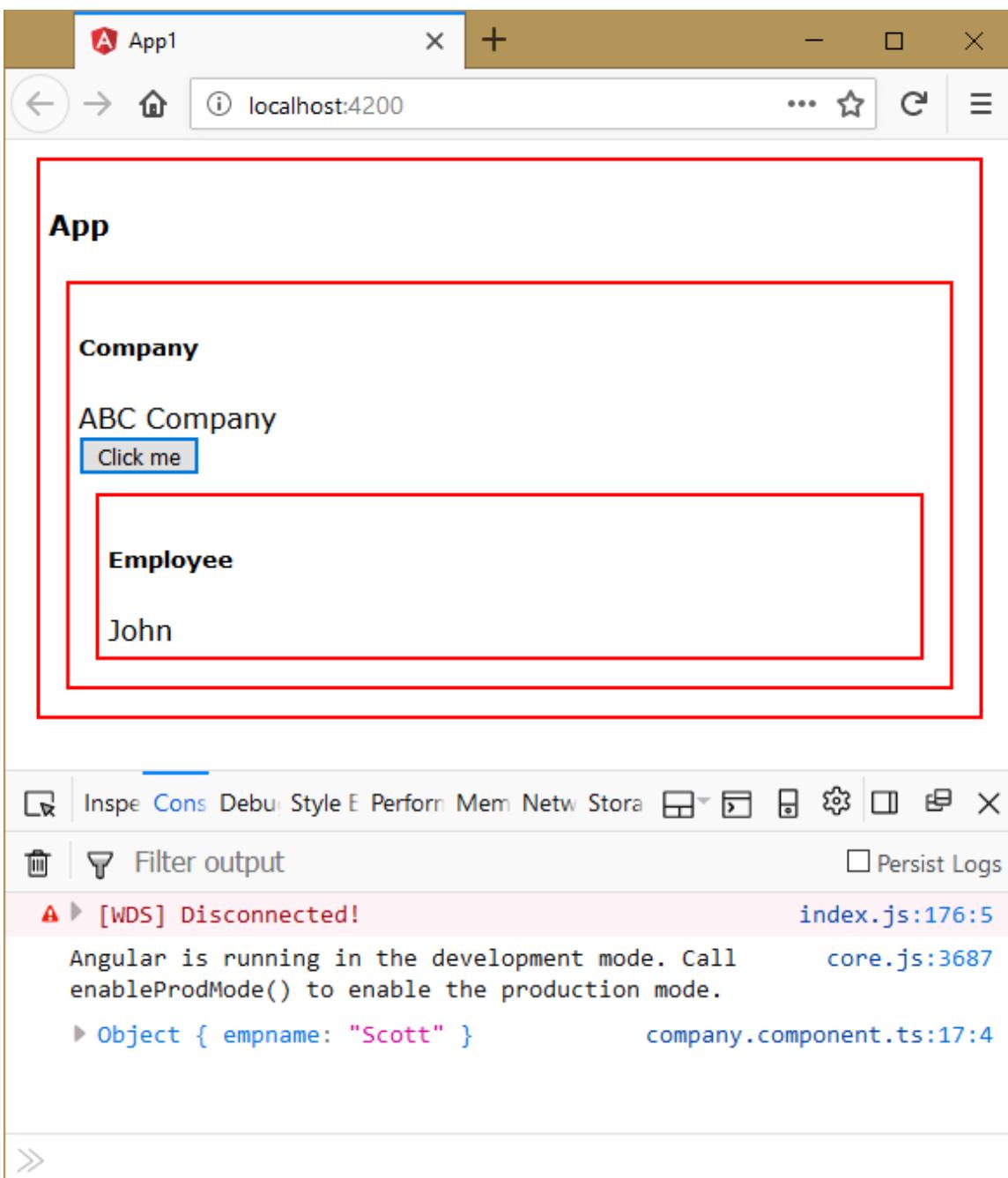
```
<div class="class1">
 <h5>Employee</h5>
 {{empname}}
</div>
```

#### Executing the application:

- Open Command Prompt and enter the following commands:  
cd c:\angular\app1  
ng serve
- Open the browser and enter the following URL:  
<http://localhost:4200>



- Click on “Click Me”.



## ViewChildren

- The “ViewChildren” represents a set of elements of specific type, which is a child of the view (template) of the component.
- ViewChildren is used to access elements, that is present in the view (template) of the component.
- ViewChildren is used to access properties / methods of the children.

### Steps:

- Import “ViewChildren” and “QueryList”:

```
import { ViewChildren, QueryList } from "@angular/core";
```

- **Create ViewChildren property:**

```
class parentcomponent
{
 @ViewChildren(classname) propertynname: QueryList<classname>;
 ...
}
```

- **Access properties / methods of the child element, using ViewChildren's property:**

```
var array = this.propertynname.toArray();
array[index].property
array[index].method()
```

### ViewChildren - Example

#### Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g component Company
ng g component Employee
```

#### c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "core-js": "2.4.1"
 }
}
```

```
"@angular/platform-browser": "^5.2.0",
"@angular/platform-browser-dynamic": "^5.2.0",
"@angular/router": "^5.2.0",
"core-js": "^2.4.1",
"rxjs": "^5.5.6",
"zone.js": "^0.8.19"
},
"devDependencies": {
"@angular/cli": "~1.7.4",
"@angular/compiler-cli": "^5.2.0",
"@angular/language-service": "^5.2.0",
"@types/jasmine": "~2.8.3",
"@types/jasminewd2": "~2.0.2",
"@types/node": "~6.0.60",
"codelyzer": "^4.0.1",
"jasmine-core": "~2.8.0",
"jasmine-spec-reporter": "~4.2.1",
"karma": "~2.0.0",
"karma-chrome-launcher": "~2.2.0",
"karma-coverage-istanbul-reporter": "^1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\styles.css

```
.class1
{
 border: 2px solid red;
 margin: 10px;
 padding: 5px;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from "@angular/forms";

import { AppComponent } from './app.component';
import { CompanyComponent } from './company/company.component';
import { EmployeeComponent } from './employee/employee.component';

@NgModule({
 declarations: [
 AppComponent,
 CompanyComponent,
```

```
 EmployeeComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

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

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{
}
```

c:\angular\app1\src\app\app.component.html

```
<div class="class1">
 <h4>App</h4>
 <app-company></app-company>
</div>
```

c:\angular\app1\src\app\company\company.component.ts

```
import { Component, ViewChildren, QueryList } from '@angular/core';
import { EmployeeComponent } from './employee/employee.component';

@Component({
 selector: 'app-company',
 templateUrl: './company.component.html',
 styleUrls: ['./company.component.css']
})
export class CompanyComponent
{
 companyname: string = "ABC Company";

 @ViewChildren(EmployeeComponent) emp: QueryList<EmployeeComponent>;

 onClickMeClicked()
 {
 console.log(this.emp);
 var a = this.emp.toArray();
 for (var i = 0; i < a.length; i++)
 {
 a[i].empname = "John";
 }
 }
}
```

```
}
```

c:\angular\app1\src\app\company\company.component.html

```
<div class="class1">
 <h5>Company</h5>
 {{companyname}}

 <input type="button" value="Click me" (click)="onClickMeClicked()">
 <app-employee></app-employee>
 <app-employee></app-employee>
</div>
```

c:\angular\app1\src\app\employee\employee.component.ts

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

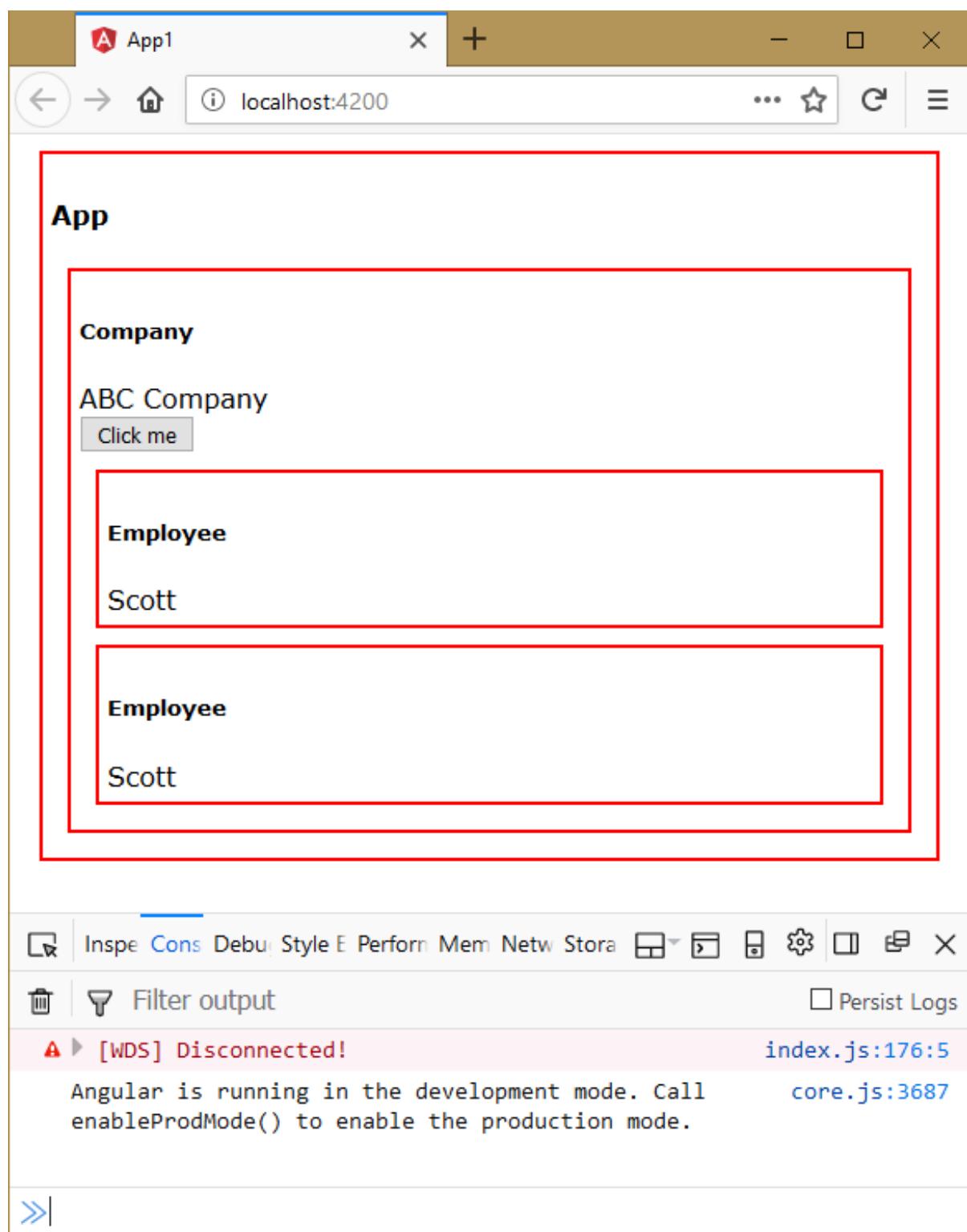
@Component({
 selector: 'app-employee',
 templateUrl: './employee.component.html',
 styleUrls: ['./employee.component.css']
})
export class EmployeeComponent {
 empname: string = "Scott";
}
```

c:\angular\app1\src\app\employee\employee.component.html

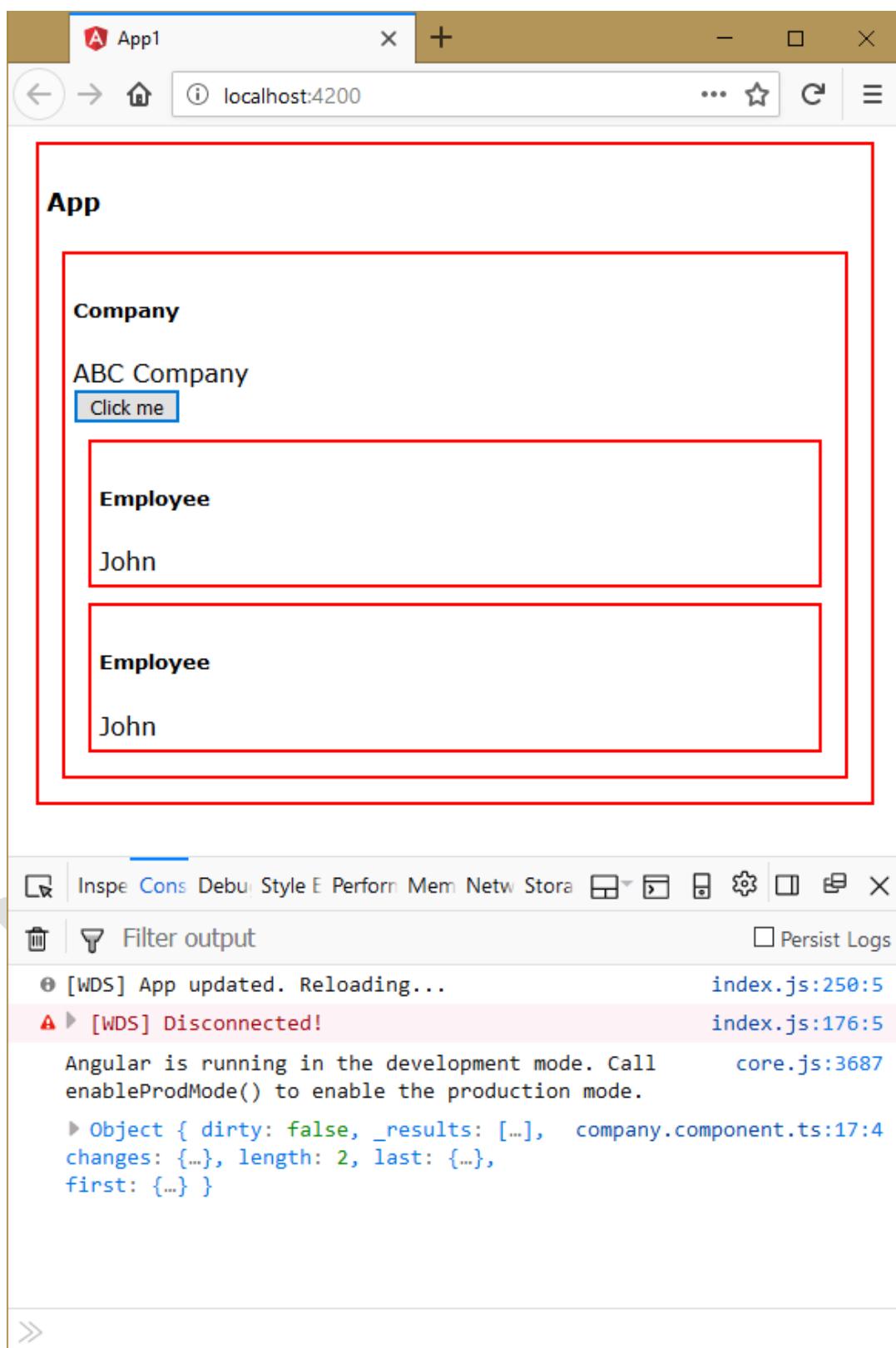
```
<div class="class1">
 <h5>Employee</h5>
 {{empname}}
</div>
```

#### Executing the application:

- Open Command Prompt and enter the following commands:  
cd c:\angular\app1  
ng serve
- Open the browser and enter the following URL:  
<http://localhost:4200>



- Click on “Click Me”.



## ContentChild

- The “ContentChild” represents an element, which is a child of the content of the component.
- ContentChild is used to access an element, that is present in the content of the component.
- ContentChild can contain a child element of a specific type (class).
- ContentChild is used to access properties / methods of the child.

### Steps:

- Import “ContentChild”:**

```
import { ContentChild } from "@angular/core";
```

- Create ContentChild property:**

```
class parentcomponent
{
 @ContentChild(classname) propertlename: classname;
 ...
}
```

- Access properties / methods of the child element, using ContentChild’s property:**

```
this.propertlename.property
this.propertlename.method()
```

## ContentChild - Example

### Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g component Company
ng g component Employee
```

### c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "postinstall": "ng build --prod"
 },
 "dependencies": {
 "@angular/common": "^7.2.1",
 "@angular/compiler": "^7.2.1",
 "@angular/core": "^7.2.1",
 "@angular/forms": "^7.2.1",
 "@angular/platform-browser": "^7.2.1",
 "@angular/platform-browser-dynamic": "^7.2.1",
 "@angular/router": "^7.2.1",
 "core-js": "^2.2.2",
 "rxjs": "^6.3.3",
 "zone.js": "^0.8.2"
```

```
"build": "ng build --prod",
"test": "ng test",
"lint": "ng lint",
"e2e": "ng e2e"
},
"private": true,
"dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
},
"devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
}
}
```

---

**c:\angular\app1\src\styles.css**

```
.class1
{
 border: 2px solid red;
 margin: 10px;
 padding: 5px;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from "@angular/forms";

import { AppComponent } from './app.component';
import { CompanyComponent } from './company/company.component';
import { EmployeeComponent } from './employee/employee.component';

@NgModule({
 declarations: [
 AppComponent,
 CompanyComponent,
 EmployeeComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

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

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div class="class1">
 <h4>App</h4>
 <app-company>
 <app-employee></app-employee>
 </app-company>
</div>
```

c:\angular\app1\src\app\company\company.component.ts

```
import { Component, ContentChild } from '@angular/core';
import { EmployeeComponent } from './employee/employee.component';

@Component({
 selector: 'app-company',
 templateUrl: './company.component.html',
 styleUrls: ['./company.component.css']
})
```

```
)
export class CompanyComponent
{
 companyname: string = "ABC Company";

 @ContentChild(EmployeeComponent) emp: EmployeeComponent;

 onClickMeClicked()
 {
 console.log(this.emp);
 this.emp.empname = "John";
 }
}
```

c:\angular\app1\src\app\company\company.component.html

```
<div class="class1">
 <h5>Company</h5>
 {{companyname}}

 <input type="button" value="Click me" (click)="onClickMeClicked()">
 <ng-content></ng-content>
</div>
```

c:\angular\app1\src\app\employee\employee.component.ts

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

@Component({
 selector: 'app-employee',
 templateUrl: './employee.component.html',
 styleUrls: ['./employee.component.css']
)
export class EmployeeComponent
{
 empname: string = "Scott";
}
```

c:\angular\app1\src\app\employee\employee.component.html

```
<div class="class1">
 <h5>Employee</h5>
 {{empname}}
</div>
```

### Executing the application:

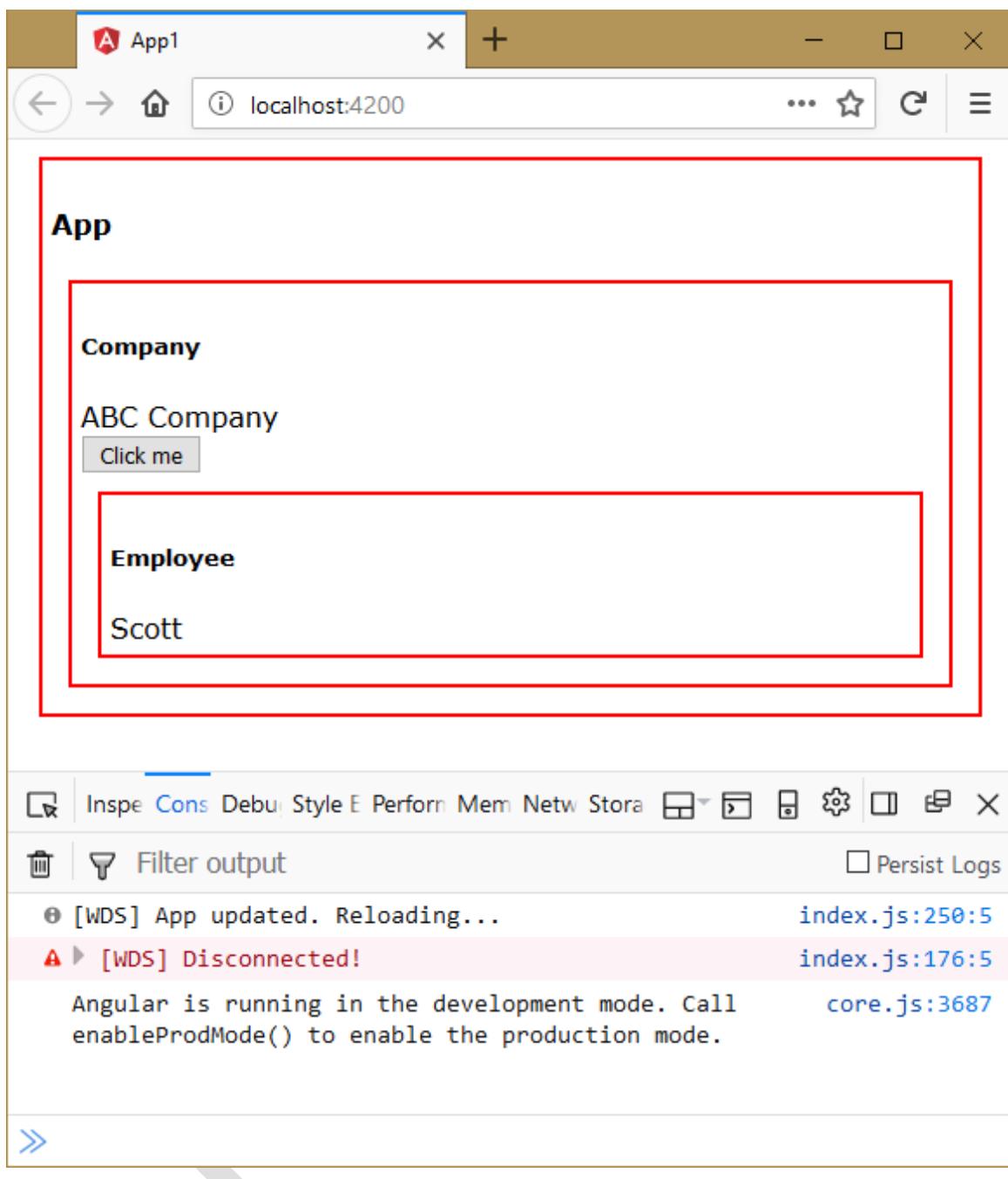
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

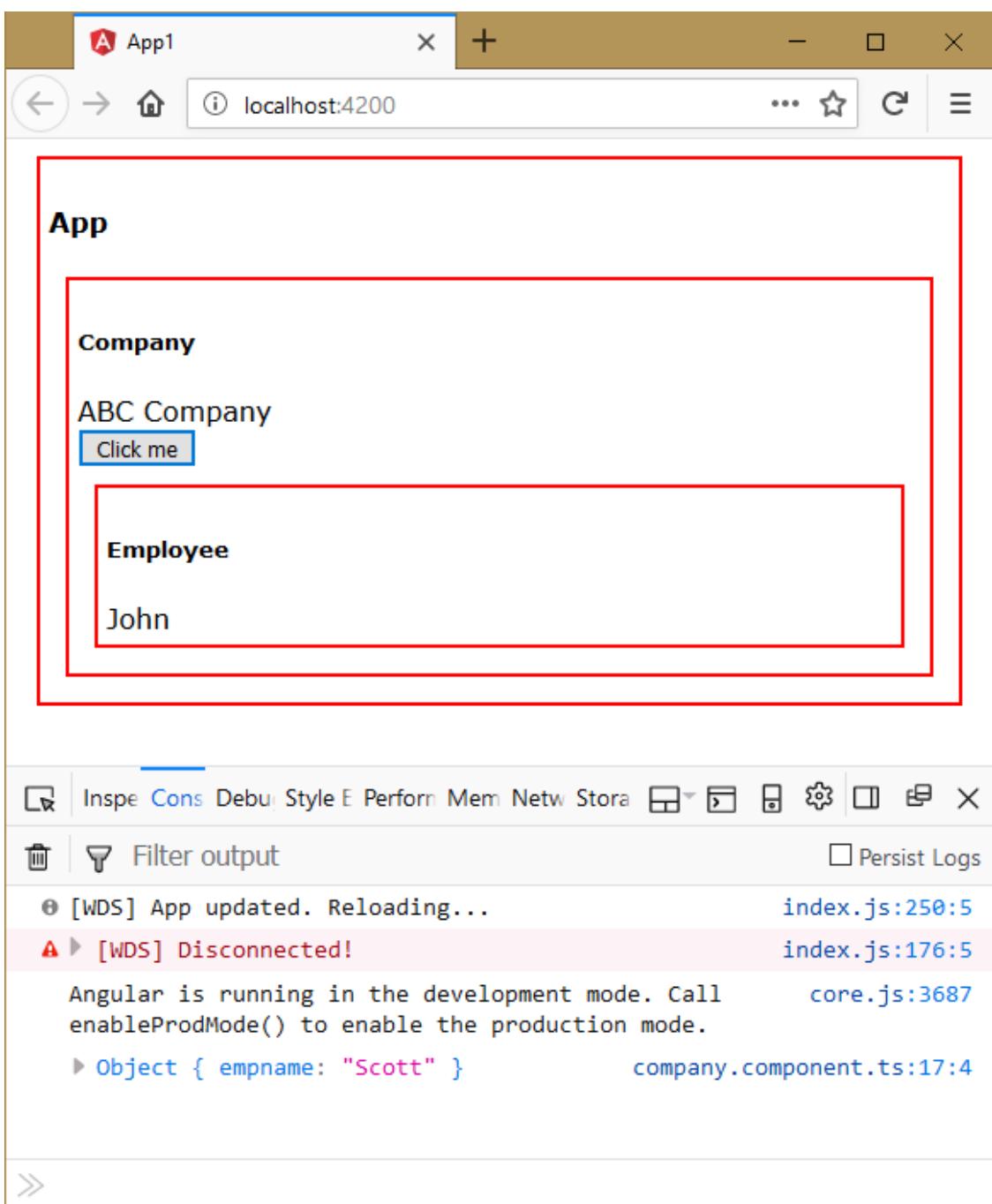
```
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



- Click on “Click Me”.



### ContentChildren

- The “ContentChildren” represents a set of elements of specific type, which is a child of the content of the component.
- ContentChildren is used to access elements, that is present in the content of the component.
- ContentChildren is used to access properties / methods of the children.

**Steps:**

- Import “ContentChildren” and “QueryList”:

```
import { ContentChildren, QueryList } from "@angular/core";
```

- Create ContentChildren property:

```
class parentcomponent
{
 @ContentChildren(classname) propertynome: QueryList<classname>;
 ...
}
```

- Access properties / methods of the child element, using ContentChildren’s property:

```
var array = this.propertynome.toArray();
array[index].property
array[index].method()
```

**ContentChildren - Example****Creating Application**

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g component Company
ng g component Employee
```

**c:\angular\app1\package.json**

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
```

```
"dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "5.5.6",
 "zone.js": "^0.8.19"
},
"devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\styles.css

```
.class1
{
 border: 2px solid red;
 margin: 10px;
 padding: 5px;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { CompanyComponent } from './company/company.component';
```

```
import { EmployeeComponent } from './employee/employee.component';

@NgModule({
 declarations: [
 AppComponent,
 CompanyComponent,
 EmployeeComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

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

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div class="class1">
<h4>App</h4>
<app-company>
<app-employee></app-employee>
<app-employee></app-employee>
</app-company>
</div>
```

c:\angular\app1\src\app\company\company.component.ts

```
import { Component, ContentChildren, QueryList } from '@angular/core';
import { EmployeeComponent } from './employee/employee.component';

@Component({
 selector: 'app-company',
 templateUrl: './company.component.html',
 styleUrls: ['./company.component.css']
})
export class CompanyComponent
{
 companyname: string = "ABC Company";

 @ContentChildren(EmployeeComponent) emp: QueryList<EmployeeComponent>;
```

```
onClickMeClicked()
{
 console.log(this.emp);
 var a = this.emp.toArray();
 for (var i = 0; i < a.length; i++)
 {
 a[i].empname = "John";
 }
}
```

c:\angular\app1\src\app\company\company.component.html

```
<div class="class1">
<h5>Company</h5>
{{companyname}}

<input type="button" value="Click me" (click)="onClickMeClicked()">
<ng-content></ng-content>
</div>
```

c:\angular\app1\src\app\employee\employee.component.ts

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

@Component({
 selector: 'app-employee',
 templateUrl: './employee.component.html',
 styleUrls: ['./employee.component.css']
})
export class EmployeeComponent
{
 empname: string = "Scott";
}
```

c:\angular\app1\src\app\employee\employee.component.html

```
<div class="class1">
<h5>Employee</h5>
{{empname}}
</div>
```

### Executing the application:

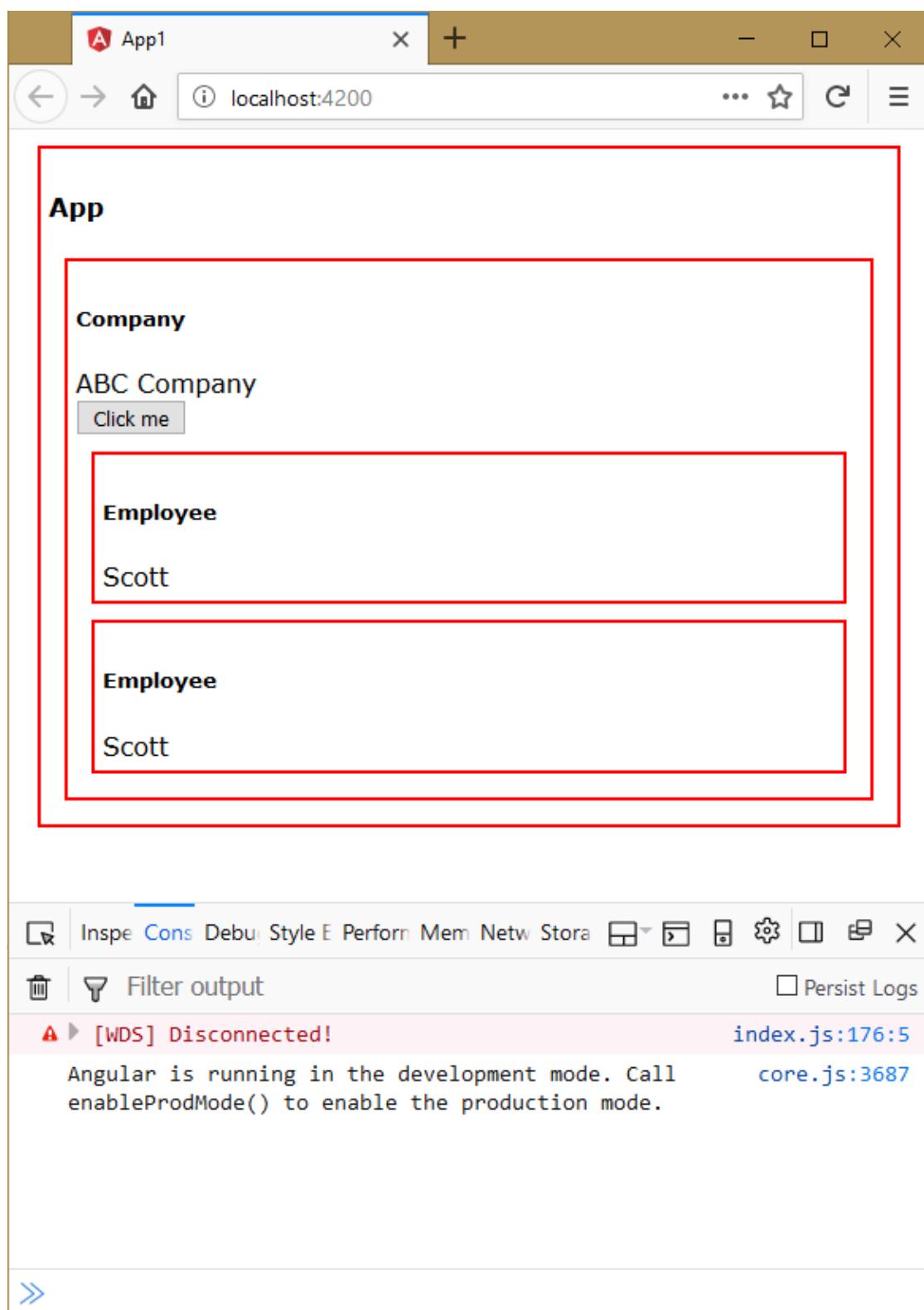
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

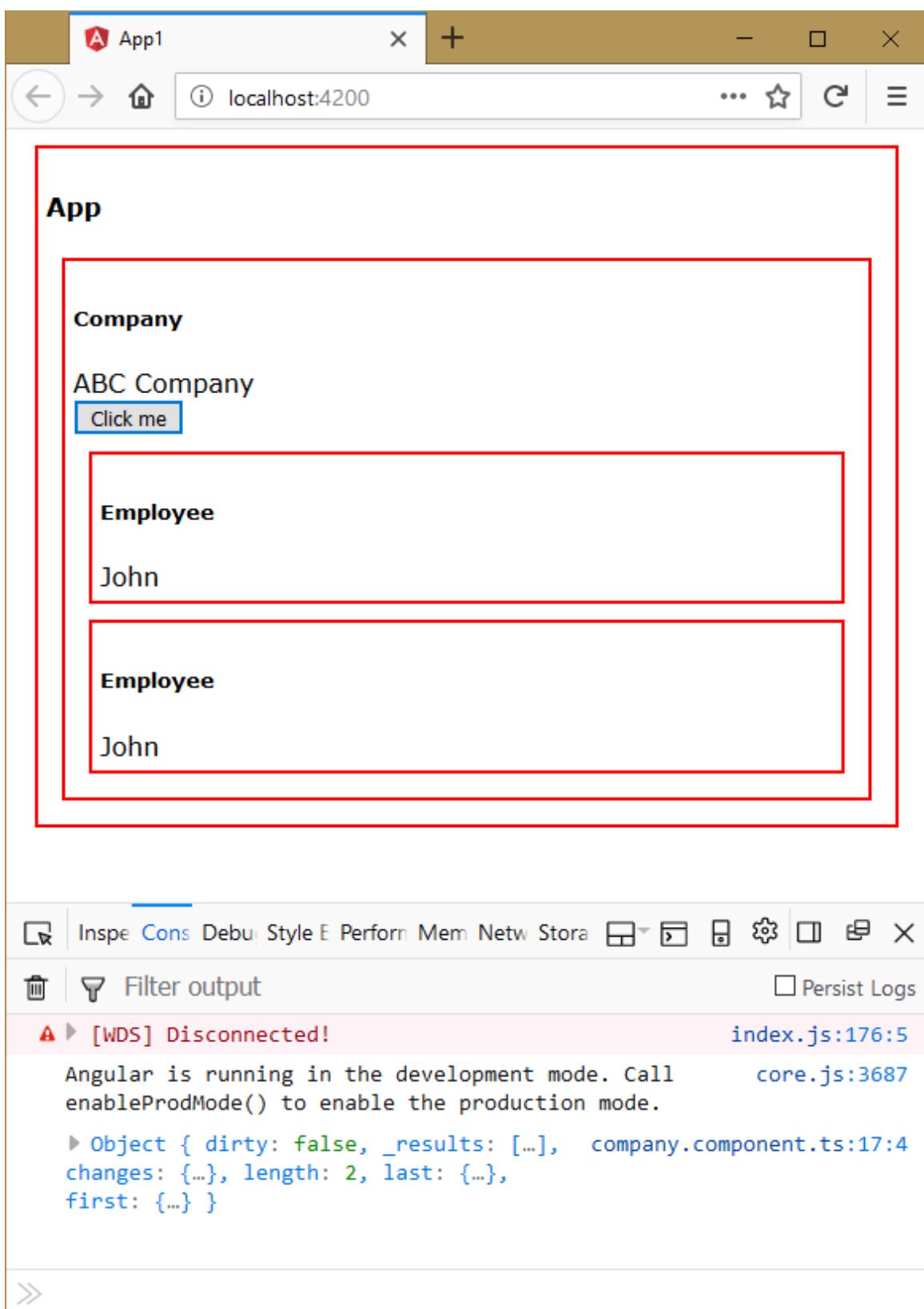
```
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



- Click on “Click Me”.



## Reference Names

- Reference names are used to access specific instance of the element in the template or content.
- You can create the reference name by writing “#” and followed by reference name in the template.
- You can access the element by specifying its reference name in the @ViewChild or @ContentChild decorator.

### Syntax:

- **Create reference name:**

```
<tag #referencename> </tag>
```

- **Access the element**

```
class parentcomponent
{
 @ViewChild("referencename") propertname: classname;
 @ContentChild("referencename") propertname: classname;
 ...
}
```

- **Access properties / methods of the child element**

```
this.propertname.property
this.propertname.method
```

## Reference Names - Example

### Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g component Company
ng g component Employee
```

### c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
```

```
"build": "ng build --prod",
"test": "ng test",
"lint": "ng lint",
"e2e": "ng e2e"
},
"private": true,
"dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
},
"devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
}
}
```

---

**c:\angular\app1\src\styles.css**

```
.class1
{
 border: 2px solid red;
 margin: 10px;
 padding: 5px;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from "@angular/forms";

import { AppComponent } from './app.component';
import { CompanyComponent } from './company/company.component';
import { EmployeeComponent } from './employee/employee.component';

@NgModule({
 declarations: [
 AppComponent,
 CompanyComponent,
 EmployeeComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

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

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div class="class1">
 <h4>App</h4>
 <app-company></app-company>
</div>
```

c:\angular\app1\src\app\company\company.component.ts

```
import { Component, ViewChild } from '@angular/core';
import { EmployeeComponent } from './employee/employee.component';

@Component({
 selector: 'app-company',
 templateUrl: './company.component.html',
 styleUrls: ['./company.component.css']
})
```

```
export class CompanyComponent
{
 companyname: string = "ABC Company";

 @ViewChild("second") emp: EmployeeComponent;

 onClickMeClicked()
 {
 console.log(this.emp);
 this.emp.empname = "John";
 }
}
```

c:\angular\app1\src\app\company\company.component.html

```
<div class="class1">
<h5>Company</h5>
{{companyname}}

<input type="button" value="Click me" (click)="onClickMeClicked()">
<app-employee></app-employee>
<app-employee #second></app-employee>
<app-employee></app-employee>
</div>
```

c:\angular\app1\src\app\employee\employee.component.ts

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

@Component({
 selector: 'app-employee',
 templateUrl: './employee.component.html',
 styleUrls: ['./employee.component.css']
})
export class EmployeeComponent
{
 empname: string = "Scott";
}
```

c:\angular\app1\src\app\employee\employee.component.html

```
<div class="class1">
<h5>Employee</h5>
{{empname}}
</div>
```

### Executing the application:

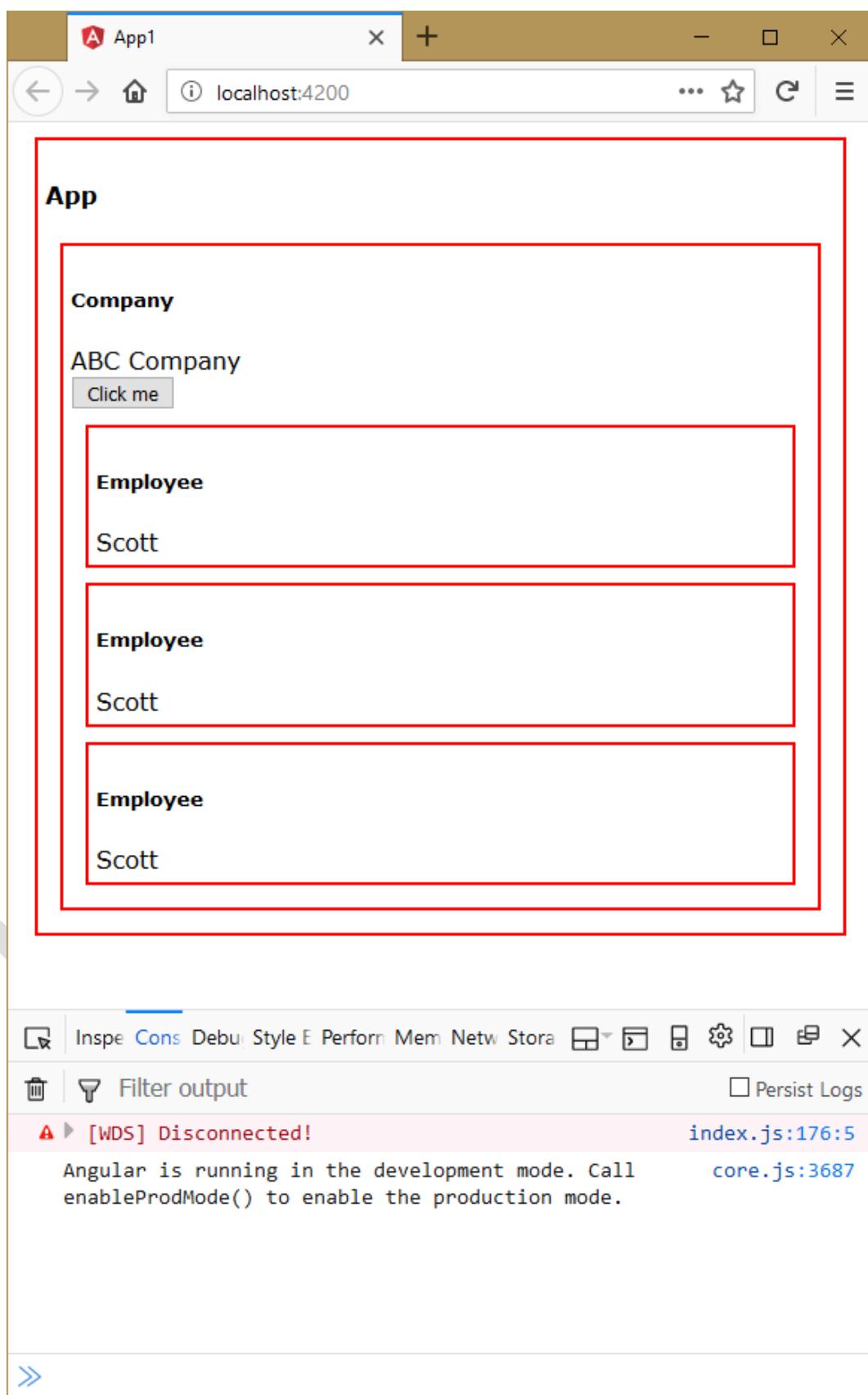
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

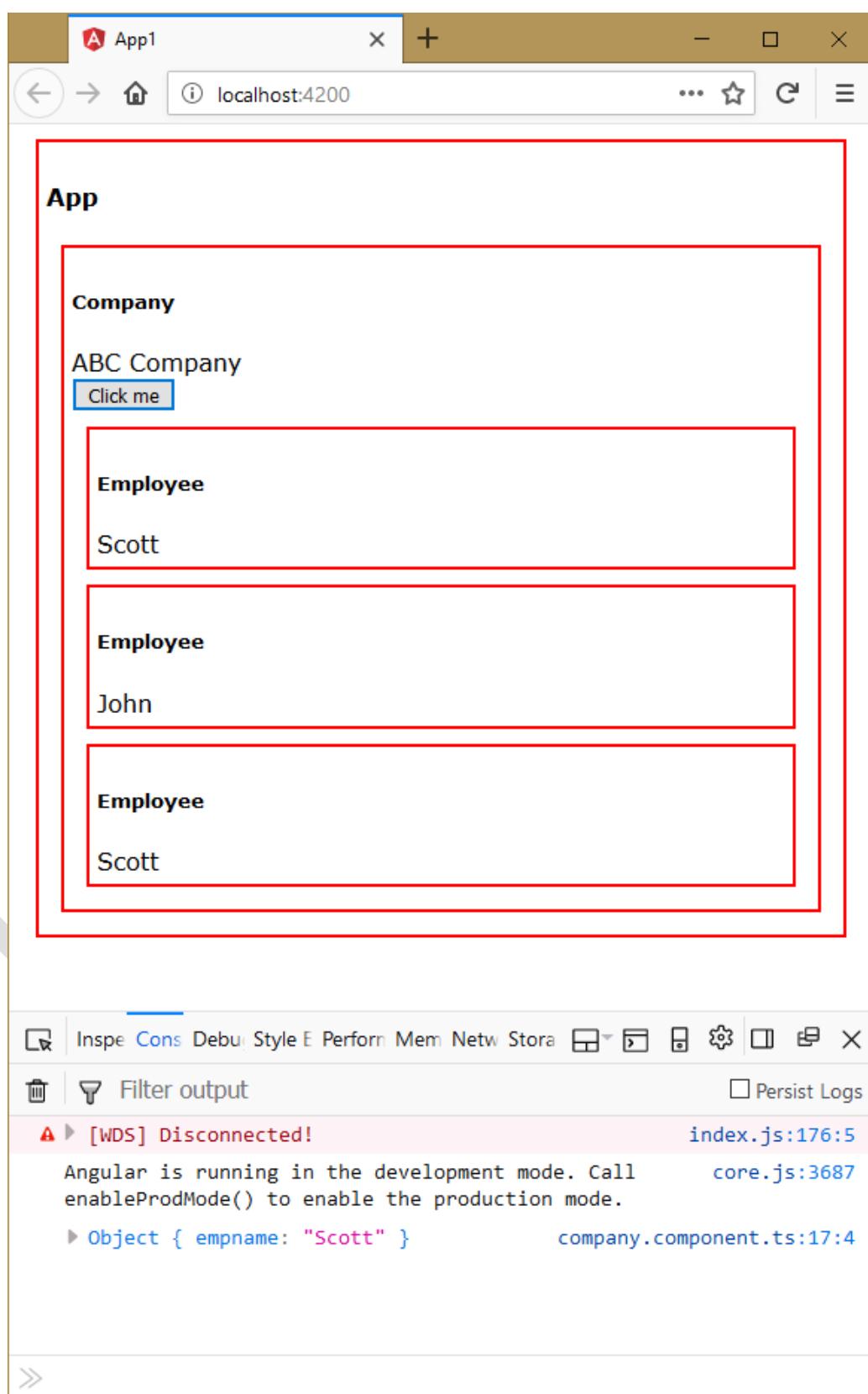
```
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



- Click on “Click Me”.



## ElementRef

- ElementRef represents a specific normal tag (not component) in the template / content.
- You can create the reference name by writing “#” and followed by reference name in the template.
- You can access the element by specifying its reference name in the @ViewChild or @ContentChild decorator.

### Syntax:

- **Create reference name:**

```
<tag #referencename> </tag>
```

- **Access the element**

```
class parentcomponent
{
 @ViewChild("referencename") propertname: ElementRef;
 @ContentChild("referencename") propertname: ElementRef;
 ...
}
```

- **Access properties / methods of the child element**

```
this.propertname.property
this.propertname.method
```

## ElementRef - Example

### Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g component Company
ng g component Employee
```

### c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
```

```
"build": "ng build --prod",
"test": "ng test",
"lint": "ng lint",
"e2e": "ng e2e"
},
"private": true,
"dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
},
"devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
}
}
```

### c:\angular\app1\src\styles.css

---

```
.class1
{
 border: 2px solid red;
 margin: 10px;
 padding: 5px;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from "@angular/forms";

import { AppComponent } from './app.component';
import { CompanyComponent } from './company/company.component';
import { EmployeeComponent } from './employee/employee.component';

@NgModule({
 declarations: [
 AppComponent,
 CompanyComponent,
 EmployeeComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

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

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div class="class1">
 <h4>App</h4>
 <app-company></app-company>
</div>
```

c:\angular\app1\src\app\company\company.component.ts

```
import { Component, ViewChild, ElementRef } from '@angular/core';

@Component({
 selector: 'app-company',
 templateUrl: './company.component.html',
 styleUrls: ['./company.component.css']
})
export class CompanyComponent
{}
```

```
companyname: string = "ABC Company";

@ViewChild("myheading") h: ElementRef;

onClickMeClicked()
{
 this.h.nativeElement.innerHTML = "New York";
}
}
```

c:\angular\app1\src\app\company\company.component.html

```
<div class="class1">
 <h5>Company</h5>
 {{companyname}}

 <input type="button" value="Click me" (click)="onClickMeClicked()">
 <h5 #myheading>Hyderabad</h5>
</div>
```

c:\angular\app1\src\app\employee\employee.component.ts

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

@Component({
 selector: 'app-employee',
 templateUrl: './employee.component.html',
 styleUrls: ['./employee.component.css']
})
export class EmployeeComponent
{
 empname: string = "Scott";
}
```

c:\angular\app1\src\app\employee\employee.component.html

```
<div class="class1">
 <h5>Employee</h5>
 {{empname}}
</div>
```

### Executing the application:

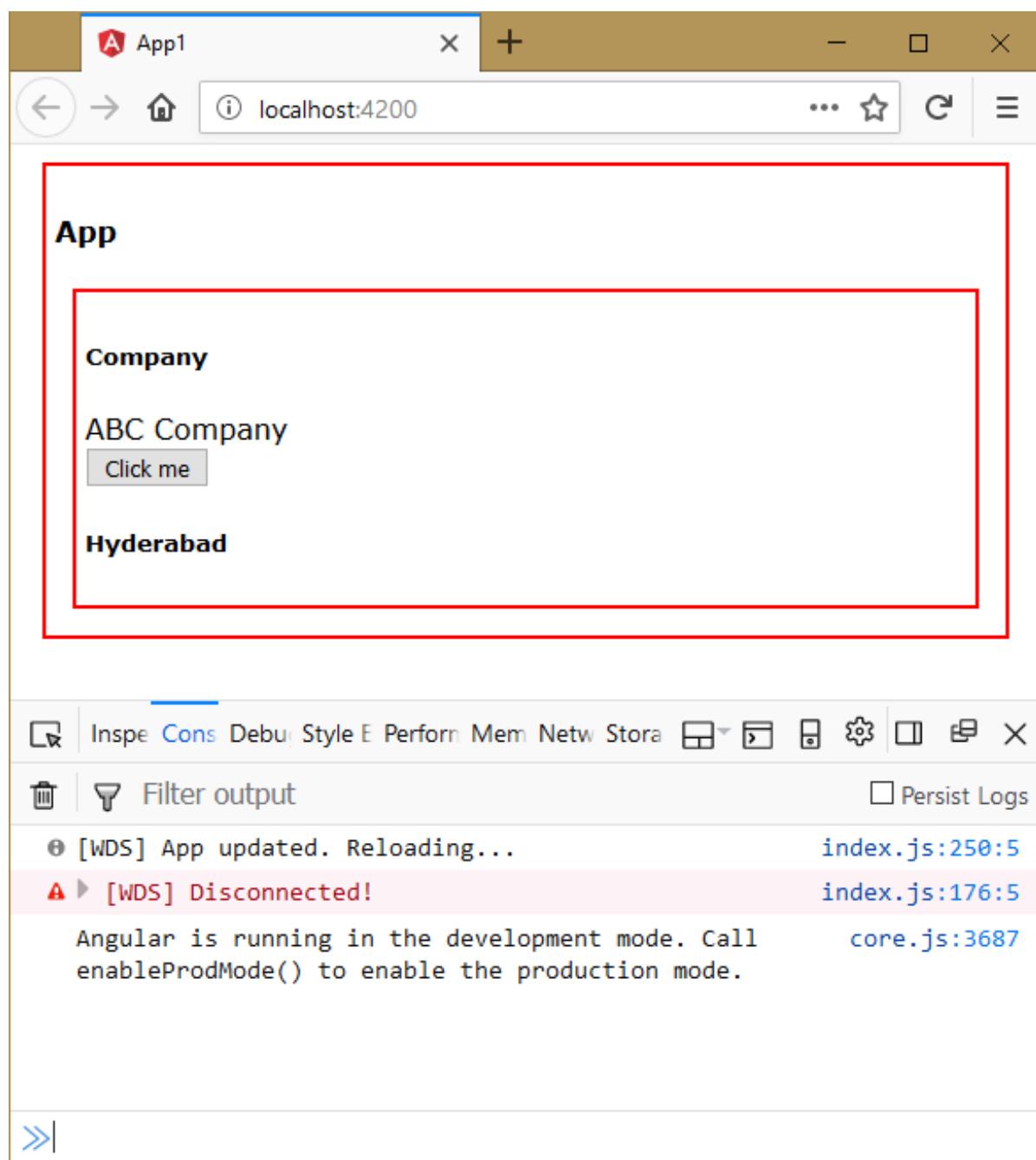
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

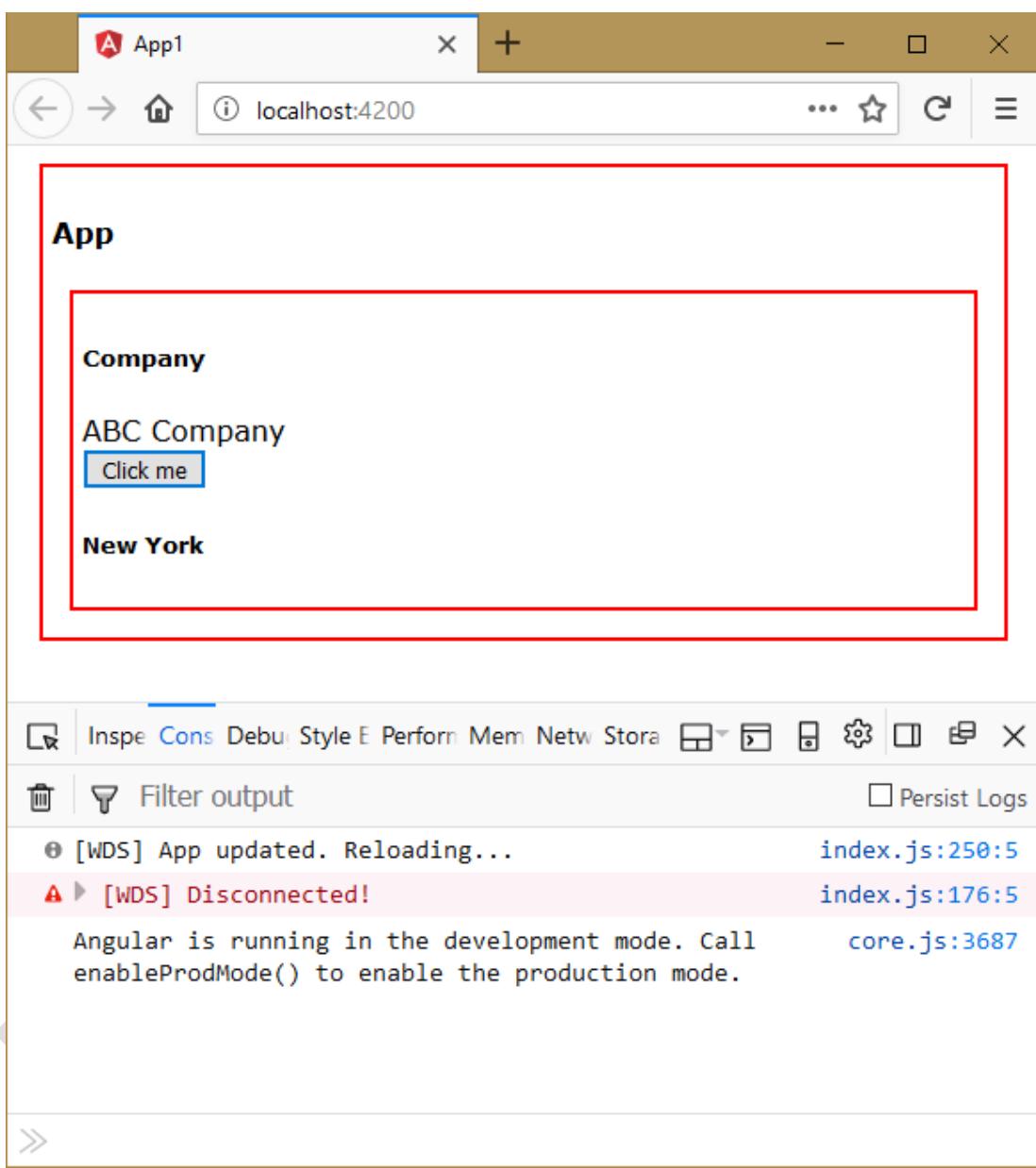
```
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```

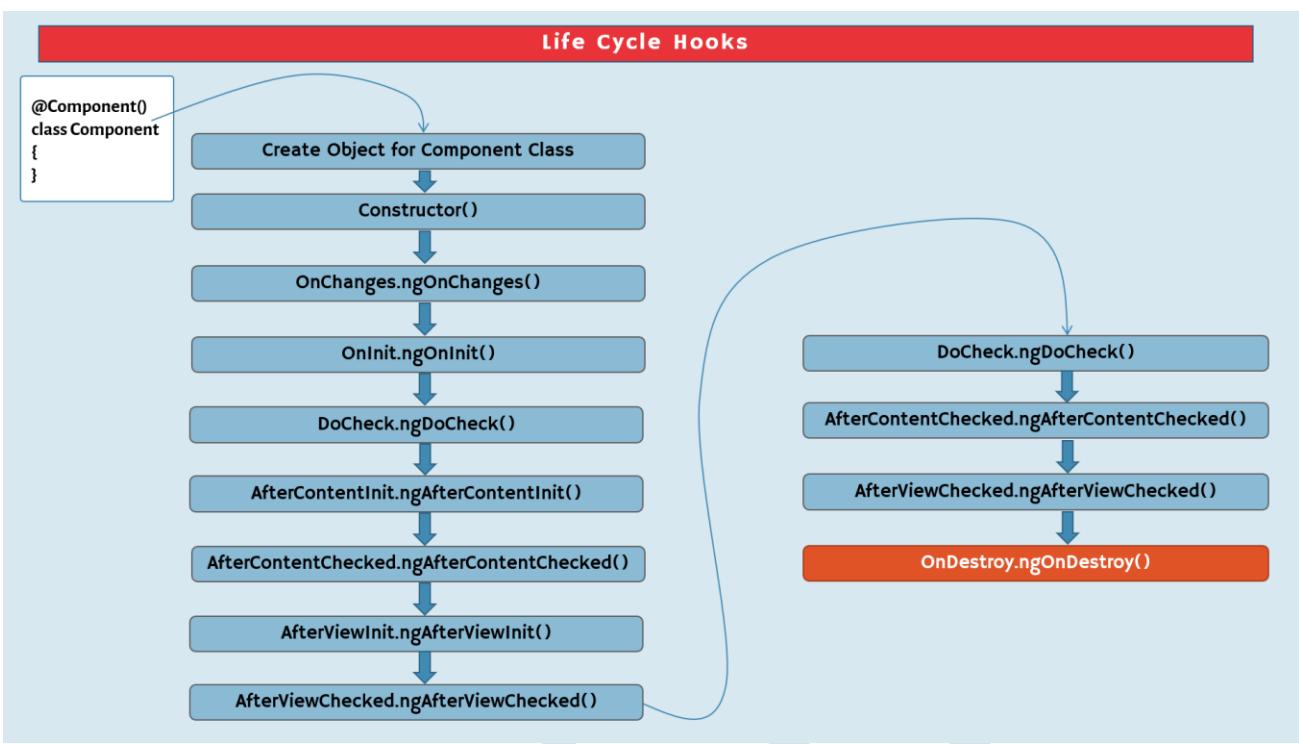


- Click on “Click Me”.



## Life Cycle Hooks

- Component has a life cycle, which is managed by Angular.
- Angular creates it, renders it, creates and renders its children, checks it when its properties changed, and destroys it before removing it from the DOM.
- Angular offers lifecycle hooks that provide visibility into these key life moments and the ability to act when they occur.
- The life cycle events will execute automatically at different stages, while executing the component.

**First run:**

1. **Component Object:** First, an object will be created for the component class. That means, the properties and methods of the component class, will be stored in the component object.
2. **Constructor:** Next, the “constructor” of component class will be executed. Use the constructor, to set default values to any properties of the component, inject services into the component.
3. **OnChanges.ngOnChanges:** Next, “ngOnChanges” method of “OnChanges” interface will be executed. This method executes when a new object is received with the new values of the input properties and just before a moment of assigning those new values into the respective input properties of the component. This method executes only if the component has input properties.
4. **OnInit.ngOnInit:** Next, “ngOnInit” method of “OnInit” interface will be executed. Use this method to call services to get data from database or any other data source.
5. **DoCheck.ngDoCheck( ):** Next, “ngDoCheck” method of “DoCheck” interface will execute. This method executes when an event occurs, such as clicking, typing some key in the board etc. Use this method to identify whether the “change detection” process occurs or not.
6. **AfterContentInit.ngAfterContentInit( ):** Next, “ngAfterContentInit” method of “AfterContentInit” interface will execute. This method executes after initializing the content of the component, which is passed while invoking the component. Use this method to set the properties of content children.
7. **AfterContentChecked.ngAfterContentChecked( ):** Next, “ngAfterContentInit” method of “AfterContentInit” interface will execute. This method executes after “change detection” process of the content is completed. Use this method to check any properties of the content children, whether those are having specific values or not.

8. **AfterViewInit.ngAfterViewInit():** Next, “ngAfterViewInit” method of “AfterViewInit” interface will execute. This method executes after initializing the view (template) of the component. Use this method to set the properties of view children.
9. **AfterViewChecked.ngAfterViewChecked():** Next, “ngAfterViewInit” method of “AfterViewInit” interface will execute. This method executes after “change detection” process of view is completed. Use this method to check any properties of the view children, whether those are having specific values or not.

### **On an event occurs:**

1. **DoCheck.ngDoCheck()**
2. **AfterContentChecked.ngAfterContentChecked()**
3. **AfterViewChecked.ngAfterViewChecked()**

### **On deleting the component:**

1. **OnDestroy.ngOnDestroy():** This method executes when the component is deleted from memory (when we close the web page in the browser).

### **Steps to handle event:**

1. Import the interface:

```
import { interfacename } from "@angular/core";
```
2. Implement the interface:

```
export class componentclassname implements interfacename
{
}
```
3. Create the method:

```
methodname()
{
 //code here
}
```

### **Life Cycle Hooks - Example**

#### **Creating Application**

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
```

```
cd c:\angular\app1
ng g component company
```

**c:\angular\app1\package.json**

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
 },
 "devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
 }
}
```

```
}
```

c:\angular\app1\src\styles.css

```
.class1
{
 border: 2px solid red;
 margin: 10px;
 padding: 5px;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from "@angular/forms";

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

@NgModule({
 declarations: [
 AppComponent,
 CompanyComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule { }
```

c:\angular\app1\src\app\app.component.ts

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

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{ }
```

c:\angular\app1\src\app\app.component.html

```
<div class="class1">
<h4>App</h4>
<app-company [companynumber]="'ABC Pvt Ltd' ">
 <h5>one</h5>
 <h5>two</h5>
```

```
<h5>three</h5>
</app-company>
</div>
```

c:\angular\app1\src\app\company\company.component.ts

```
import { Component, Input, OnChanges, OnInit, DoCheck, AfterContentInit, AfterContentChecked, AfterViewInit, AfterViewChecked } from '@angular/core';

@Component({
 selector: 'app-company',
 templateUrl: './company.component.html',
 styleUrls: ['./company.component.css']
})
export class CompanyComponent implements OnInit, DoCheck, AfterContentInit, AfterContentChecked, AfterViewInit, AfterViewChecked {
 @Input() companyname: string;

 constructor()
 {
 console.log("constructor");
 }

 ngOnInit()
 {
 console.log("ngOnInit");
 }

 ngOnChanges()
 {
 console.log("ngOnChanges");
 }

 ngDoCheck()
 {
 console.log("ngDoCheck");
 }

 ngAfterContentInit()
 {
 console.log("ngAfterContentInit");
 }

 ngAfterContentChecked()
 {
 console.log("ngAfterContentChecked");
 }

 ngAfterViewInit()
 {
 console.log("ngAfterViewInit");
 }
}
```

```
ngAfterViewChecked()
{
 console.log("ngAfterViewChecked");
}

ngOnDestroy()
{
 console.log("ngOnDestroy");
}
}
```

c:\angular\app1\src\app\company\company.component.html

```
<div class="class1">
<h5>Company</h5>
<input type="text" [(ngModel)]="companynname">
<ng-content></ng-content>
</div>
```

#### Executing the application:

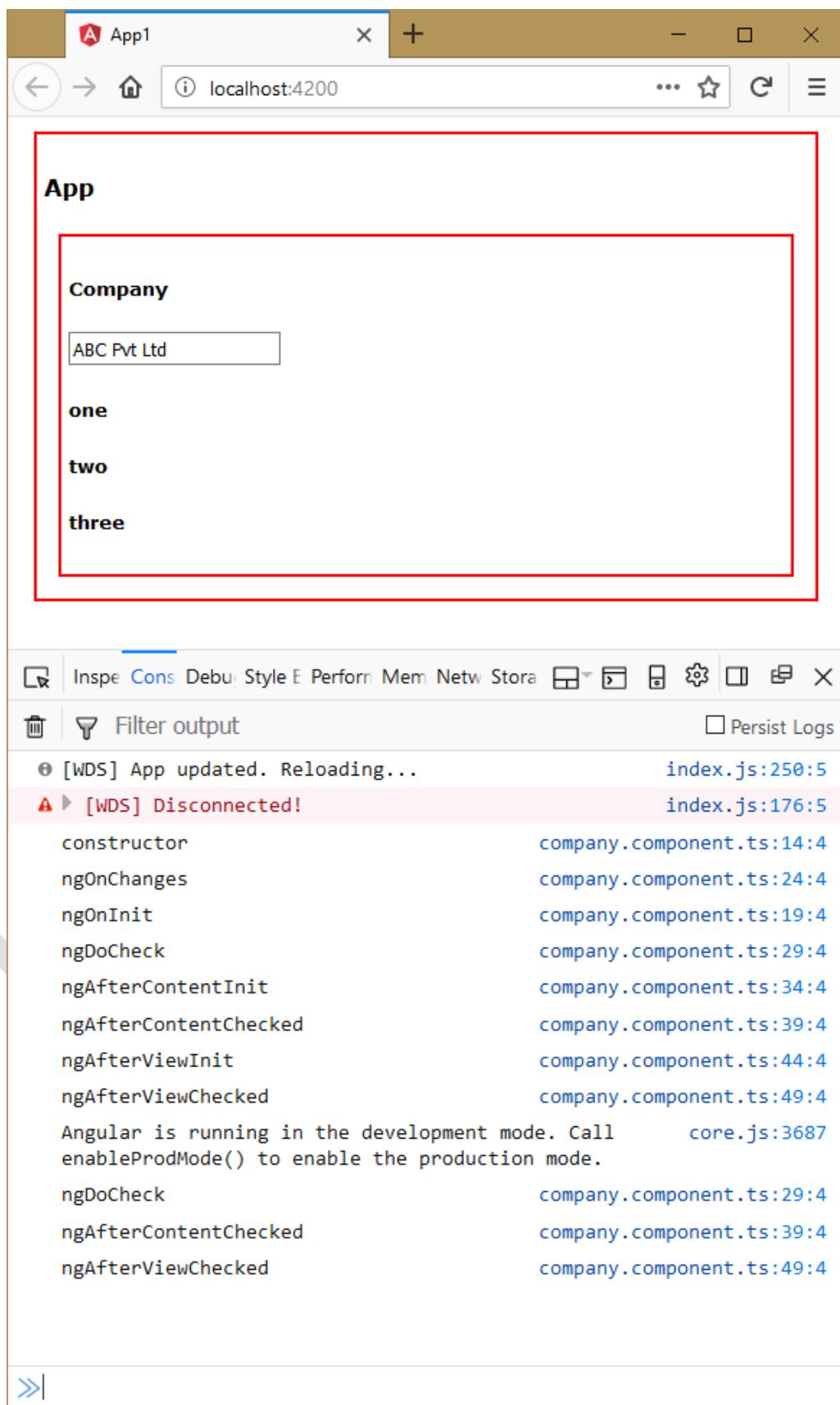
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

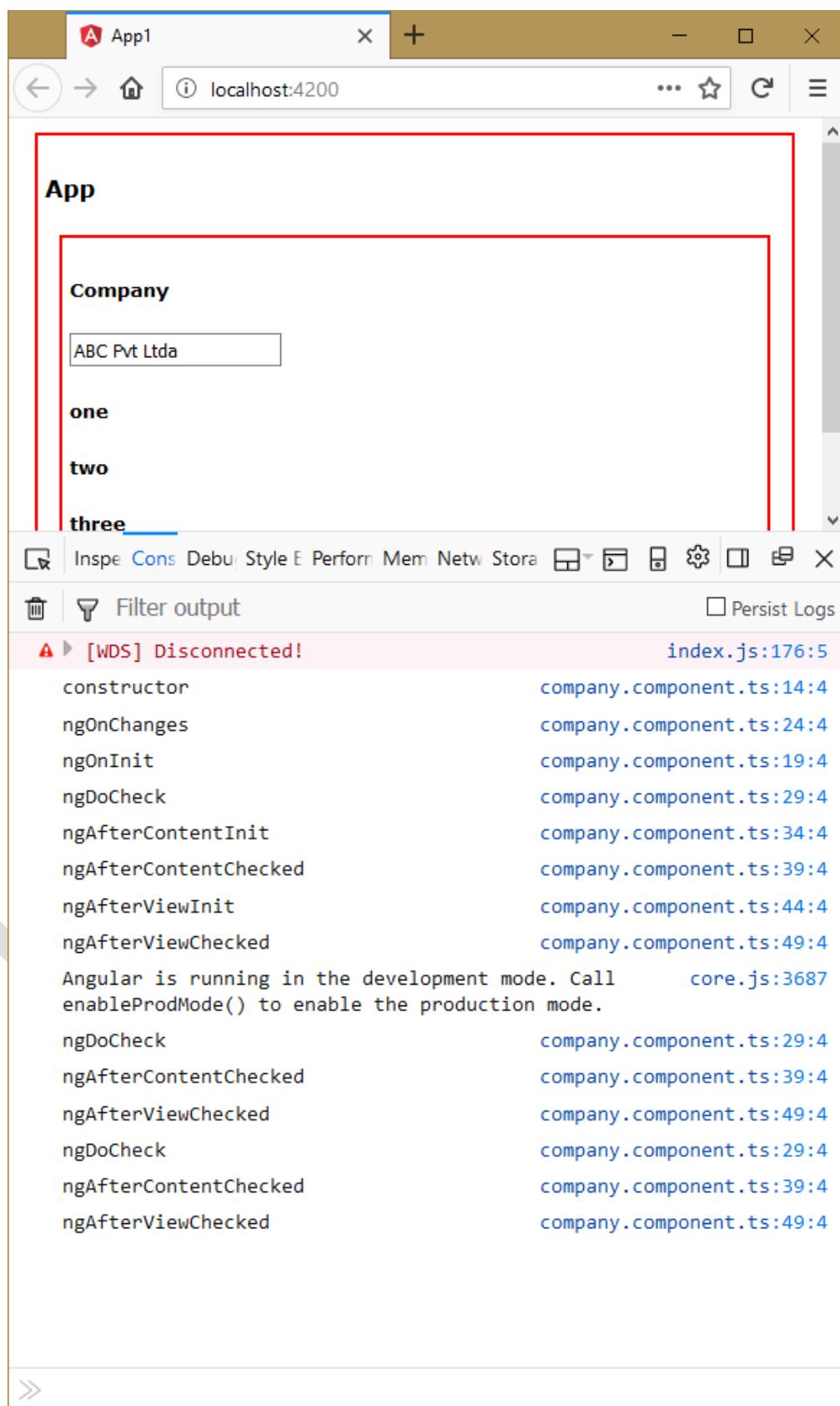
```
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```

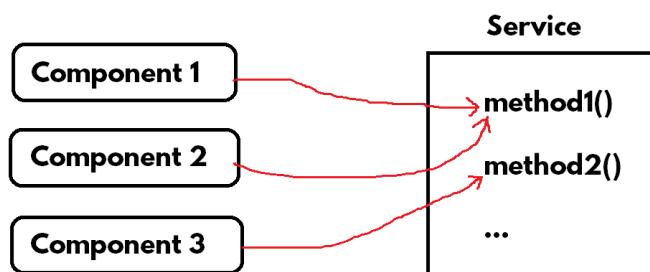


After typing some character in the textbox:



## Services

- The service is a class contains re-usable code (business logic, validations, calculations etc.), which can be called in one or more components. If you place the re-usable set of properties and methods as a service class, it can be called from any component or any other service in the entire application.
- We must decorate the service class with “@Injectable()” decorator, to make the service accessible from any other component. You can import “@Injectable” decorator from “@angular/core” package.
- We must use “@Inject()” decorator, to request angular to create an object for the service class. Then the angular framework will automatically creates an object for the service class and passes the object as argument for your component’s constructor; you can receive it into a reference variable in the constructor. You can use “@Inject” only in the constructor of component. To make the reference variable as member of the component class, add “private” or “public” keyword at left side of the reference variable in the constructor.
- In realtime, all the CRUD operations (Ajax calls) are created in the service; the same is called in the component class, whenever required.



### Steps to handle event:

- Create Service:**

```

import { Injectable } from "@angular/core";
@Injectable()
class Serviceclassname
{
 Methods here
}

```

- Add service as provider in the component:**

```

@Component({ ..., providers: [Serviceclassname] })
class Componentclassname
{
}

```

- **(or) Add service as provider in the module:**

```
@NgModule({ providers: [Serviceclassname] })
class ModuleClassname
{
}
}
```

- **Get the instance of service using dependency injection:**

```
import { Inject } from "@angular/core";

@Component({ ... })
class ComponentClassname
{
 constructor(@Inject(Serviceclassname) variable : Serviceclassname)
 {
 }
}
```

## Services - Example

### Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g class User
ng g service Login
```

### c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 }
}
```

```
},
"private": true,
"dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
},
"devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';
import { LoginService } from './login.service';

@NgModule({
 declarations: [
 AppComponent
],
 imports: [
 BrowserModule, FormsModule
],
})
```

```
 providers: [LoginService],
 bootstrap: [AppComponent]
 })
export class AppModule { }
```

c:\angular\app1\src\app\user.ts

```
export class User {
 username: string;
 password: string;

 constructor(a: string, b: string)
 {
 this.username = a;
 this.password = b;
 }
}
```

c:\angular\app1\src\app\login.service.ts

```
import { Injectable } from '@angular/core';
import { User } from './user';

@Injectable()
export class LoginService {

 users: User[] = [
 new User("scott", "scott123"),
 new User("smith", "smith123"),
 new User("allen", "allen123"),
];
 constructor() {}

 checkUsernameAndPassword(username: string, password: string): boolean
 {
 var count = 0;
 for (var i = 0; i < this.users.length; i++)
 {
 if (this.users[i].username == username && this.users[i].password == password)
 {
 count++;
 }
 }
 if (count == 1)
 {
 return true;
 }
 else
 {
 return false;
 }
 }
}
```

c:\angular\app1\src\app\app.component.ts

```

import { Component, Inject } from "@angular/core";
import { LoginService } from "./login.service";

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{
 username: string = "";
 password: string = "";
 msg: string = "";

 constructor(@Inject(LoginService) private s : LoginService)
 {
 }

 CheckLogin(txt1)
 {
 if (this.s.checkUsernameAndPassword(this.username, this.password) == true)
 {
 this.msg = "Successful login";
 }
 else
 {
 this.msg = "Invalid login";
 txt1.focus();
 }
 }
}

```

c:\angular\app1\src\app\app.component.html

```

<div>
 <form>
 <h4>Services</h4>
 Username: <input type="text" [(ngModel)]="username" name="username" #t1>

 Password: <input type="password" [(ngModel)]="password" name="password">

 <input type="submit" value="Login" (click)="CheckLogin(t1)">

 {{msg}}
 </form>
</div>

```

### Executing the application:

- Open Command Prompt and enter the following commands:

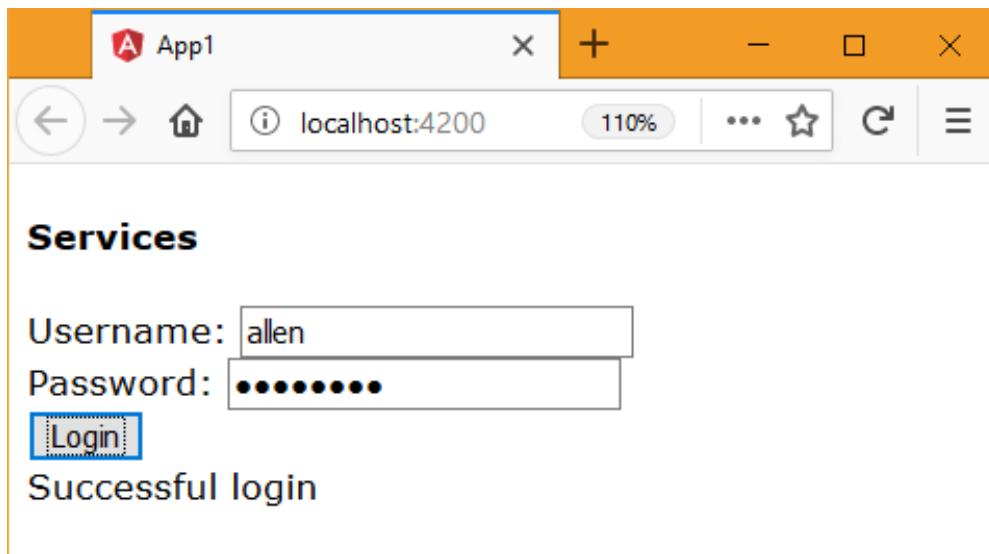
```

cd c:\angular\app1
ng serve

```

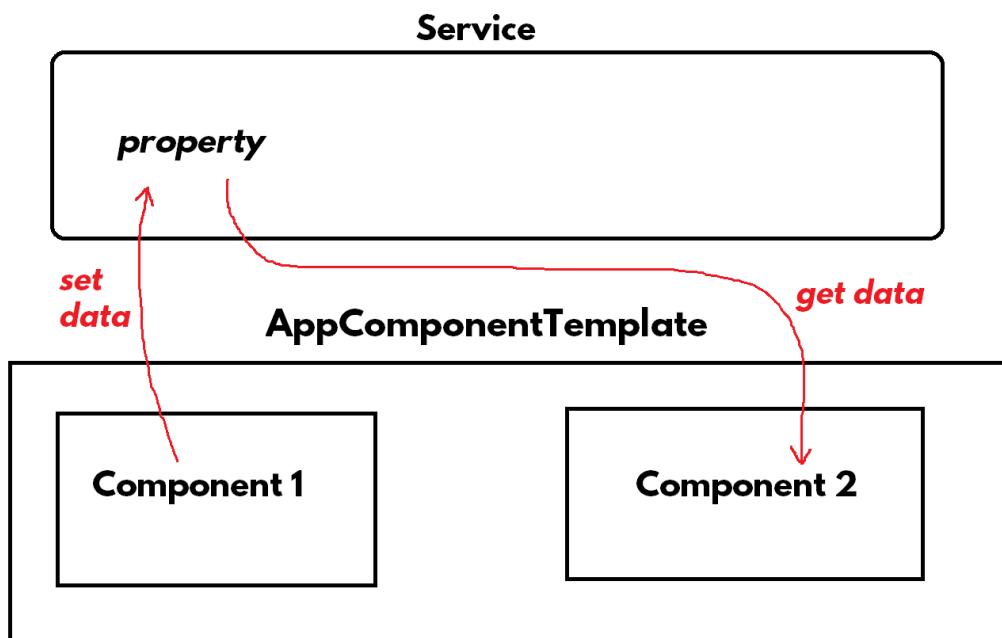
- Open the browser and enter the following URL:

http://localhost:4200



### Sharing Data using Services

- We can't share data among sibling components directly; but we can do it by using service.
- We can set data from component1 to service; Then the component2 can access data from service.



### Sharing Data using Services - Example

#### Creating Application

- Open Command Prompt and enter the following commands:
- ```
cd c:\angular
ng new app1
```

```
cd c:\angular\app1  
ng g component India  
ng g component Usa  
ng g service Population
```

c:\angular\app1\package.json

```
{  
  "name": "app1",  
  "version": "0.0.0",  
  "license": "MIT",  
  "scripts": {  
    "ng": "ng",  
    "start": "ng serve",  
    "build": "ng build --prod",  
    "test": "ng test",  
    "lint": "ng lint",  
    "e2e": "ng e2e"  
  },  
  "private": true,  
  "dependencies": {  
    "@angular/animations": "^5.2.0",  
    "@angular/common": "^5.2.0",  
    "@angular/compiler": "^5.2.0",  
    "@angular/core": "^5.2.0",  
    "@angular/forms": "^5.2.0",  
    "@angular/http": "^5.2.0",  
    "@angular/platform-browser": "^5.2.0",  
    "@angular/platform-browser-dynamic": "^5.2.0",  
    "@angular/router": "^5.2.0",  
    "core-js": "^2.4.1",  
    "rxjs": "^5.5.6",  
    "zone.js": "^0.8.19"  
  },  
  "devDependencies": {  
    "@angular/cli": "~1.7.4",  
    "@angular/compiler-cli": "^5.2.0",  
    "@angular/language-service": "^5.2.0",  
    "@types/jasmine": "~2.8.3",  
    "@types/jasminewd2": "~2.0.2",  
    "@types/node": "~6.0.60",  
    "codelyzer": "^4.0.1",  
    "jasmine-core": "~2.8.0",  
    "jasmine-spec-reporter": "~4.2.1",  
    "karma": "~2.0.0",  
    "karma-chrome-launcher": "~2.2.0",  
    "karma-coverage-istanbul-reporter": "^1.2.1",  
    "karma-jasmine": "~1.1.0",  
    "karma-jasmine-html-reporter": "^0.2.2",  
    "protractor": "~5.1.2",  
    "ts-node": "~4.1.0",  
  }  
}
```

```
    "tslint": "~5.9.1",
    "typescript": "~2.5.3"
  }
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";
import { IndiaComponent } from './india/india.component';
import { UsaComponent } from './usa/usa.component';
import { PopulationService } from "./population.service";

@NgModule({
  declarations: [
    AppComponent,
    IndiaComponent,
    UsaComponent
  ],
  imports: [
    BrowserModule, FormsModule
  ],
  providers: [ PopulationService ],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\population.service.ts

```
import { Injectable } from '@angular/core';

@Injectable()
export class PopulationService {
  IndiaPopulation: number;
}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div>
```

```
<h4>Sharing Data using Services</h4>
<app-india></app-india>
<app-usa></app-usa>
</div>
```

c:\angular\app1\src\app\india\india.component.ts

```
import { Component, OnInit, Inject } from '@angular/core';
import { PopulationService } from './population.service';

@Component({
  selector: 'app-india',
  templateUrl: './india.component.html',
  styleUrls: ['./india.component.css']
})
export class IndiaComponent implements OnInit
{
  population: number = 1.32;

  constructor(@Inject(PopulationService) private ps : PopulationService)
  {
    this.ps.IndiaPopulation = this.population;
  }

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\india\india.component.html

```
<div class="class1">
<h4>India</h4>
</div>
```

c:\angular\app1\src\app\usa\Usa.component.ts

```
import { Component, OnInit, Inject } from '@angular/core';
import { PopulationService } from './population.service';

@Component({
  selector: 'app-usa',
  templateUrl: './usa.component.html',
  styleUrls: ['./usa.component.css']
})
export class UsaComponent implements OnInit
{
  indiapopulation: number = 1.32;

  constructor(@Inject(PopulationService) private ps : PopulationService)
  {
    this.indiapopulation = this.ps.IndiaPopulation;
  }
}
```

```
ngOnInit()
{
}
}
```

c:\angular\app1\src\app\usa\Usa.component.html

```
<div class="class1">
<h4>United States</h4>
India population:{{indiapopulation}} billion.
</div>
```

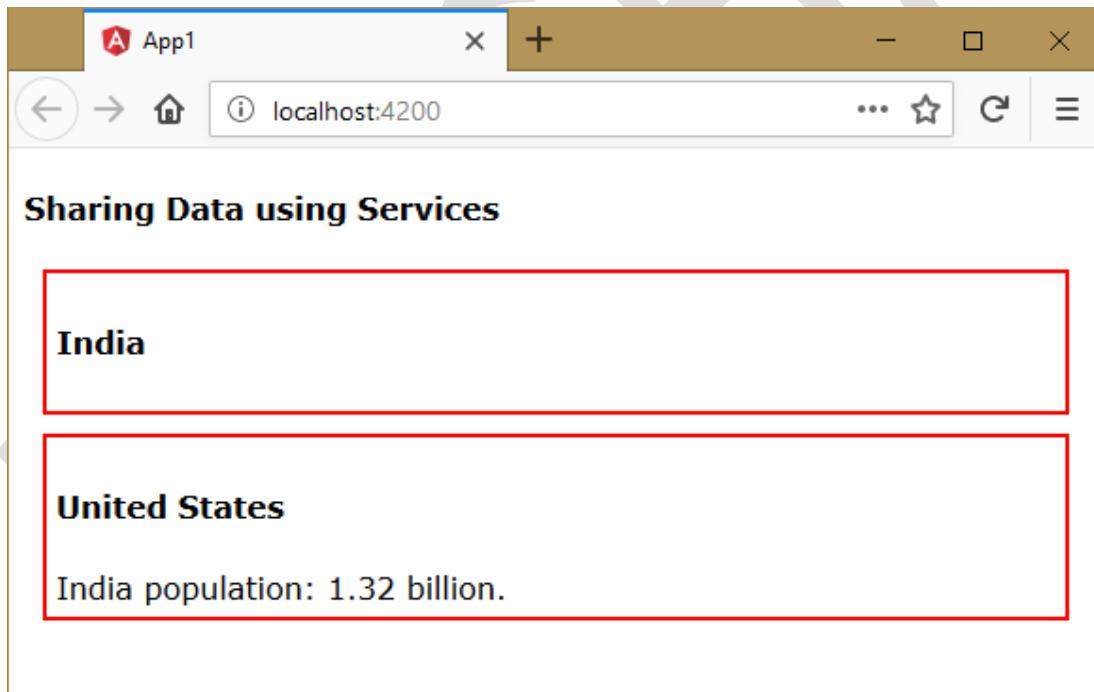
Executing the application:

- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



Custom Directives

- Directive is a class, that can be invoked (called) through an attribute of a tag in the template.
- Directive provides additional functionality for the html element.
- For example, "ngIf" directive checks the condition, displays the element if the condition is TRUE; and removes the element if the condition is false.
- The " ElementRef" class represents the element, in which the directive is invoked.
- Directive can receive values from the element using @Input() decorator.
- Directive can add events to the element by using @HostListener() decorator.

- We can communicate between the component to the directive, using @ViewChild decorator in the component.

Steps for Working with Directives

- **Create directive:**

```
@Directive({ selector: "[directiveattributename]" })  
class directiveclassname  
{  
  constructor(@Inject(ElementRef) referencename : ElementRef)  
  {  
  }  
  
  @Input() directiveproperty : datatype;  
  
  @HostListener("eventname")  
  methodname()  
  {  
  }  
}
```

- **Add directive to the module:**

```
@NgModule({ ..., declarations: [ ..., directiveclassname ] })  
class moduleclassname  
{  
}
```

- **Invoke directive from html tag:**

```
<tag directiveattributename directiveproperty=" value "> </tag>
```

Custom Directives - Example

Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular  
ng new app1  
cd c:\angular\app1
```

ng g directive Sample

c:\angular\app1\package.json

```
{  
  "name": "app1",  
  "version": "0.0.0",  
  "license": "MIT",  
  "scripts": {  
    "ng": "ng",  
    "start": "ng serve",  
    "build": "ng build --prod",  
    "test": "ng test",  
    "lint": "ng lint",  
    "e2e": "ng e2e"  
  },  
  "private": true,  
  "dependencies": {  
    "@angular/animations": "^5.2.0",  
    "@angular/common": "^5.2.0",  
    "@angular/compiler": "^5.2.0",  
    "@angular/core": "^5.2.0",  
    "@angular/forms": "^5.2.0",  
    "@angular/http": "^5.2.0",  
    "@angular/platform-browser": "^5.2.0",  
    "@angular/platform-browser-dynamic": "^5.2.0",  
    "@angular/router": "^5.2.0",  
    "core-js": "^2.4.1",  
    "rxjs": "^5.5.6",  
    "zone.js": "^0.8.19"  
  },  
  "devDependencies": {  
    "@angular/cli": "~1.7.4",  
    "@angular/compiler-cli": "^5.2.0",  
    "@angular/language-service": "^5.2.0",  
    "@types/jasmine": "~2.8.3",  
    "@types/jasminewd2": "~2.0.2",  
    "@types/node": "~6.0.60",  
    "codelyzer": "^4.0.1",  
    "jasmine-core": "~2.8.0",  
    "jasmine-spec-reporter": "~4.2.1",  
    "karma": "~2.0.0",  
    "karma-chrome-launcher": "~2.2.0",  
    "karma-coverage-istanbul-reporter": "^1.2.1",  
    "karma-jasmine": "~1.1.0",  
    "karma-jasmine-html-reporter": "^0.2.2",  
    "protractor": "~5.1.2",  
    "ts-node": "~4.1.0",  
    "tslint": "~5.9.1",  
    "typescript": "~2.5.3"  
  }  
}
```

c:\angular\app1\src\styles.css

```
.class1
{
  background-color: #91e3e9;
  width: 500px;
  height: 300px;
  margin: 20px;
  float: left;
}

.class2
{
  background-color: #28abdd;
  width: 200px;
  height: 100px;
  margin: 20px;
  float: left;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";
import { SampleDirective } from './sample.directive';

@NgModule({
  declarations: [
    AppComponent,
    SampleDirective
  ],
  imports: [
    BrowserModule, FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div>
  <h4>Custom Directives</h4>
  <img width="30px" appSample firstimage="assets/tick1.jpg" secondimage="assets/tick2.jpg">
</div>
```

c:\angular\app1\src\app\sample.directive.ts

```
import { Directive, Inject, ElementRef, Input, HostListener, OnChanges } from '@angular/core'

@Directive({
  selector: '[appSample]'
})
export class SampleDirective implements OnChanges
{
  constructor( @Inject(ElementRef) private element: ElementRef)
  {
    this.element.nativeElement.style.border = "1px solid blue";
  }

  ngOnChanges()
  {
    this.element.nativeElement.setAttribute("src", this.secondimage);
  }

  @Input() firstimage: string;
  @Input() secondimage: string;

  @HostListener("mouseover")
  onMouseOver()
  {
    this.element.nativeElement.setAttribute("src", this.firstimage);
  }

  @HostListener("mouseout")
  onMouseOut()
  {
    this.element.nativeElement.setAttribute("src", this.secondimage);
  }
}
```

Executing the application:

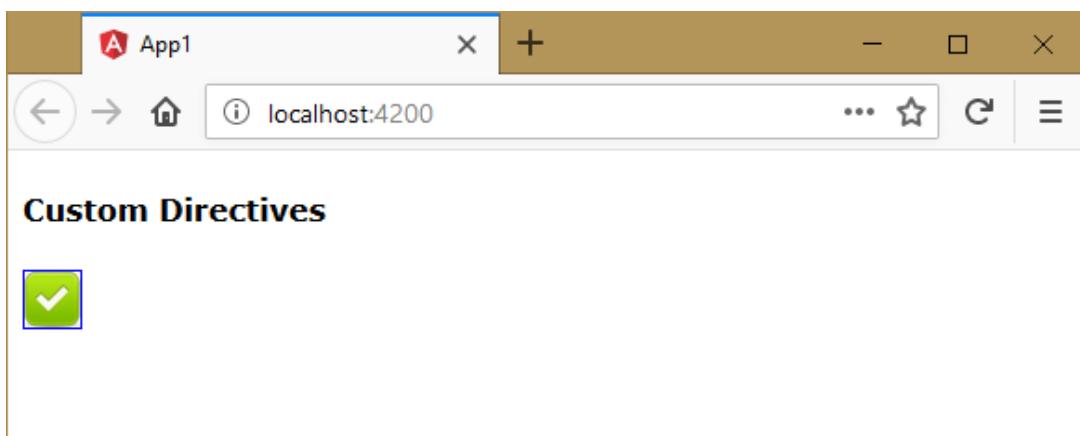
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

- Open the browser and enter the following URL:

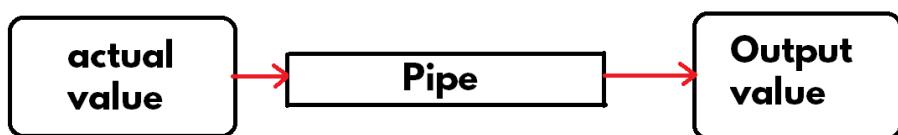
```
http://localhost:4200
```



Pipes

- Pipes transform the value into “user-expected format”.
- Pipes are invoked in expressions (interpolation binding), through pipe (|) symbol.

Syntax: {{ property | pipe }}



List of Built-in Pipes in Angular 2+

1. uppercase
2. lowercase
3. slice
4. number
5. currency
6. percent
7. date
8. json

Pipes - Example

Creating Application

- Open Command Prompt and enter the following commands:

```

cd c:\angular
ng new app1
cd c:\angular\app1
  
```

c:\angular\app1\package.json

```
{
  "name": "app1",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build --prod",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^5.2.0",
    "@angular/common": "^5.2.0",
    "@angular/compiler": "^5.2.0",
    "@angular/core": "^5.2.0",
    "@angular/forms": "^5.2.0",
    "@angular/http": "^5.2.0",
    "@angular/platform-browser": "^5.2.0",
    "@angular/platform-browser-dynamic": "^5.2.0",
    "@angular/router": "^5.2.0",
    "core-js": "^2.4.1",
    "rxjs": "^5.5.6",
    "zone.js": "^0.8.19"
  },
  "devDependencies": {
    "@angular/cli": "~1.7.4",
    "@angular/compiler-cli": "^5.2.0",
    "@angular/language-service": "^5.2.0",
    "@types/jasmine": "~2.8.3",
    "@types/jasminewd2": "~2.0.2",
    "@types/node": "~6.0.60",
    "codemlyzer": "~4.0.1",
    "jasmine-core": "~2.8.0",
    "jasmine-spec-reporter": "~4.2.1",
    "karma": "~2.0.0",
    "karma-chrome-launcher": "~2.2.0",
    "karma-coverage-istanbul-reporter": "^1.2.1",
    "karma-jasmine": "~1.1.0",
    "karma-jasmine-html-reporter": "^0.2.2",
    "protractor": "~5.1.2",
    "ts-node": "~4.1.0",
    "tslint": "~5.9.1",
    "typescript": "~2.5.3"
  }
}
```

c:\angular\app1\src\styles.css

```
.class1
{
  border: 2px solid red;
  margin: 20px;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule, FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{
  city: string = "Hyderabad";
  salary: number = 752487500;
  n: number = 0.72;
  person: object = { firstname: "Adam", lastname: "Smith" };
  dt: Date = new Date();
}
```

c:\angular\app1\src\app\app.component.html

```
<div class="class1">
<h4>Pipes</h4>
City: {{city}}<br>
Salary: {{salary}}<br>
n: {{n}}<br>
Current time: {{dt}}<br>
Person: {{person}}<br>
<hr>
```

```
Uppercase: {{city | uppercase}}<br>
Lowercase: {{city | lowercase}}<br>
Slice: {{city | slice: 2: 6}}<br>
number: {{salary | number}}<br>
number .2: {{salary | number: ".2"}}<br>
currency: {{salary | currency: "USD"}}<br>
INR: {{salary | currency: "INR"}}<br>
GBP: {{salary | currency: "GBP"}}<br>
EUR: {{salary | currency: "EUR"}}<br>
n (percent): {{n | percent}}<br>
json: {{person | json}}<br>
Short date: {{dt | date: "shortDate"}}<br>
Medium date: {{dt | date: "mediumDate"}}<br>
Long date: {{dt | date: "longDate"}}<br>
Full date: {{dt | date: "fullDate"}}<br>
Short time: {{dt | date: "shortTime"}}<br>
Long time: {{dt | date: "mediumTime"}}<br>
Short: {{dt | date: "short"}}<br>
Medium: {{dt | date: "medium"}}<br>
d/M/y: {{dt | date: "d/M/y"}}<br>
y-M-d: {{dt | date: "y-M-d"}}<br>
h:m:s: {{dt | date: "h:m:s"}}<br>
H:m: {{dt | date: "H:m"}}<br>
a: {{dt | date: "a"}}<br>
EEE: {{dt | date: "EEE"}}<br>
EEEE: {{dt | date: "EEEE"}}<br>
MMM: {{dt | date: "MMM"}}<br>
MMMM: {{dt | date: "MMMM"}}<br>
Z: {{dt | date: "Z"}}
</div>
```

Executing the application:

- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```

Pipes

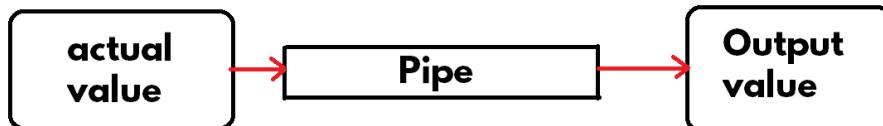
City: Hyderabad
Salary: 752487500
n: 0.72
Current time: Sun Apr 22 2018 21:54:56 GMT+0530 (India Standard Time)
Person: [object Object]

Uppercase: HYDERABAD
Lowercase: hyderabad
Slice: dera
number: 752,487,500
number .2: 752,487,500.00
currency: \$752,487,500.00
INR: ₹752,487,500.00
GBP: £752,487,500.00
EUR: €752,487,500.00
n (percent): 72%
json: { "firstname": "Adam", "lastname": "Smith" }
Short date: 4/22/18
Medium date: Apr 22, 2018
Long date: April 22, 2018
Full date: Sunday, April 22, 2018
Short time: 9:54 PM
Long time: 9:54:56 PM
Short: 4/22/18, 9:54 PM
Medium: Apr 22, 2018, 9:54:56 PM
d/M/y: 22/4/2018
y-M-d: 2018-4-22
h:m:s: 9:54:56
H:m: 21:54
a: PM
EEE: Sun
EEEE: Sunday
MMM: Apr
MMMM: April
Z: +0530

Custom Pipes

- Custom pipes are the user-defined pipes.
- Custom pipe must be a class that has `@Pipe()` decorator and implements “`PipeTransform`” interface.
- The “`PipeTransform`” interface has “`transform`” method, which must be implemented in your pipe class.
- The “`transform`” method will be executed automatically, when the pipe is invoked in the expression (through pipe `(|)` symbol).
- The “`transform`” method receive the input value as argument, do process, and return the result value, which will be displayed in the output.

Syntax to call pipe: `{{ property | pipe }}`



Syntax of Custom Pipe Class

```

import { Pipe, PipeTransform } from "@angular/core";
@Pipe({ name: "namehere" })
class custompipeclassname implements PipeTransform
{
    transform(value: datatype) : returndatatype
    {
        //do something the value here
        return (modified value);
    }
}
  
```

Add Pipe to the module

```

@NgModule( { ..., declarations: [ custompipeclassname ], ... })
class moduleclassname
{
}
  
```

Invoke the pipe in the template

`{{componentproperty | pipename}}`

Custom Pipes - Example

Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g pipe Duration
```

c:\angular\app1\package.json

```
{
  "name": "app1",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build --prod",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^5.2.0",
    "@angular/common": "^5.2.0",
    "@angular/compiler": "^5.2.0",
    "@angular/core": "^5.2.0",
    "@angular/forms": "^5.2.0",
    "@angular/http": "^5.2.0",
    "@angular/platform-browser": "^5.2.0",
    "@angular/platform-browser-dynamic": "^5.2.0",
    "@angular/router": "^5.2.0",
    "core-js": "^2.4.1",
    "rxjs": "^5.5.6",
    "zone.js": "^0.8.19"
  },
  "devDependencies": {
    "@angular/cli": "~1.7.4",
    "@angular/compiler-cli": "^5.2.0",
    "@angular/language-service": "^5.2.0",
    "@types/jasmine": "~2.8.3",
    "@types/jasminewd2": "~2.0.2",
    "@types/node": "~6.0.60",
    "codelyzer": "^4.0.1",
    "jasmine-core": "~2.8.0",
    "jasmine-spec-reporter": "~4.2.1",
    "karma": "~2.0.0",
    "karma-chrome-launcher": "~2.2.0",
    "protractor": "~5.4.0"
  }
}
```

```
"karma-coverage-istanbul-reporter": "^1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\styles.css

```
.class1
{
  border: 2px solid red;
  margin: 20px;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";
import { DurationPipe } from './duration.pipe';

@NgModule({
  declarations: [
    AppComponent,
    DurationPipe
  ],
  imports: [
    BrowserModule, FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{
  videoduration: number = 150;
}
```

c:\angular\app1\src\app\app.component.html

```
<div class="class1">
  <h4>Custom Pipes</h4>
  Video Duration: {{videoduration | duration: 'hoursandminutes' }}
</div>
```

c:\angular\app1\src\app\duration.pipe.ts

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'duration'
})
export class DurationPipe implements PipeTransform {
  transform(value: number, format: string): string {
    var s;
    if (format == "hoursonly")
      s = value / 60 + " hrs";
    else if (format == "hoursandminutes")
      s = Math.floor(value / 60) + " hrs " + ((value % 60 > 0) ? (" and " + value % 60) + " mins" : "");
    else
      s = value;
    return s;
  }
}
```

Executing the application:

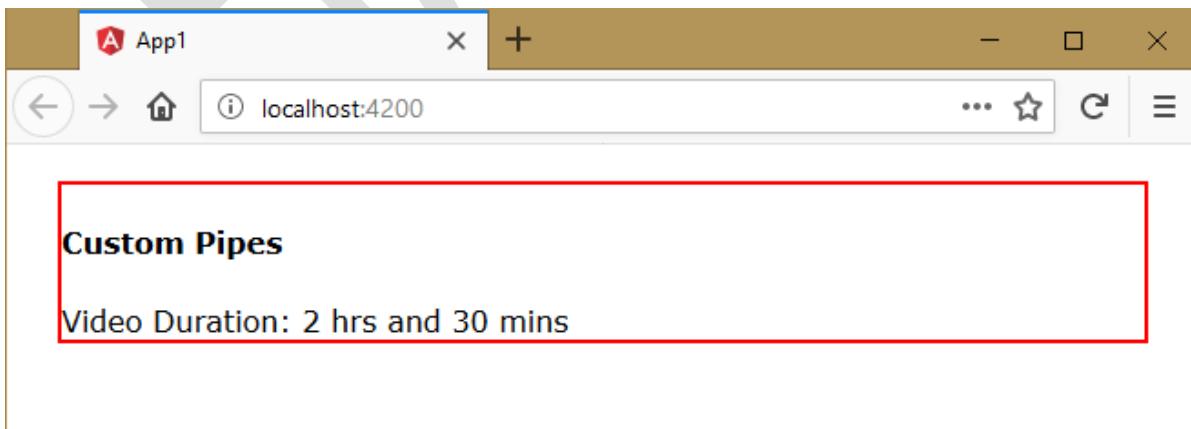
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



Forms and Validations

Template Driven Forms

- Template Driven Forms are suitable for development of simple forms with limited no. of fields and simple validations.
- In these forms, each field is represented as a property in the component class.
- Validations rules are defined in the template, using “html 5” attributes. Validation messages are displayed using “validation properties” of angular.
- “FormsModule” should be imported from “@angular/forms” package.

HTML 5 attributes for validations:

- required="required" : Field is mandatory
- minlength="n" : Minimum no. of characters
- pattern="reg exp" : Regular expression

Validation Properties:

- untouched
 - true : Field is not focused.
 - false : Field is focused.
- touched
 - true : Field is focused.
 - false : Field is not focused.
- pristine
 - true : Field is not modified by the user.
 - false : Field is modified by the user.
- dirty
 - true : Field is modified by the user.
 - false : Field is not modified by the user.
- valid
 - true : Field value is valid.
 - false : Field value is invalid
- invalid
 - true : Field value is invalid.
 - false : Field value is valid.
- errors : Represents the list of errors of the field.
 - required : true / false
 - minlength : true / false

- pattern : true / false
- number : true / false
- email : true / false
- url : true / false

| Sl. No | Description | Regular Expression |
|--------|---|--|
| 1 | Digits only | ^[0-9]*\$ |
| 2 | Alphabets only | ^[a-zA-Z]*\$ |
| 3 | Indian Mobile Number | ^[789]\d{9}\$ |
| 4 | Email | \w+([-.\']\w+)*@\w+([-.\']\w+)*\.\w+([-.\']\w+)* |
| 5 | Usernames: Alphabets, Digits and Hyphens only | ([A-Za-z0-9-]+) |
| 6 | Passwords: 6 to 15 characters; atleast one upper case letter, one lower case letter and one digit | ((?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,15}) |

Template Driven Forms - Example

- We are going to create a sample template driven form with validations.

Fields:

- Firstname
- Lastname
- Email
- Amount
- Gender
- Country

Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
```

c:\angular\app1\package.json

```
{
  "name": "app1",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build --prod",
```

```
"test": "ng test",
"lint": "ng lint",
"e2e": "ng e2e"
},
"private": true,
"dependencies": {
  "@angular/animations": "^5.2.0",
  "@angular/common": "^5.2.0",
  "@angular/compiler": "^5.2.0",
  "@angular/core": "^5.2.0",
  "@angular/forms": "^5.2.0",
  "@angular/http": "^5.2.0",
  "@angular/platform-browser": "^5.2.0",
  "@angular/platform-browser-dynamic": "^5.2.0",
  "@angular/router": "^5.2.0",
  "core-js": "^2.4.1",
  "rxjs": "^5.5.6",
  "zone.js": "^0.8.19"
},
"devDependencies": {
  "@angular/cli": "~1.7.4",
  "@angular/compiler-cli": "^5.2.0",
  "@angular/language-service": "^5.2.0",
  "@types/jasmine": "~2.8.3",
  "@types/jasminewd2": "~2.0.2",
  "@types/node": "~6.0.60",
  "codelyzer": "^4.0.1",
  "jasmine-core": "~2.8.0",
  "jasmine-spec-reporter": "~4.2.1",
  "karma": "~2.0.0",
  "karma-chrome-launcher": "~2.2.0",
  "karma-coverage-istanbul-reporter": "^1.2.1",
  "karma-jasmine": "~1.1.0",
  "karma-jasmine-html-reporter": "^0.2.2",
  "protractor": "~5.1.2",
  "ts-node": "~4.1.0",
  "tslint": "~5.9.1",
  "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\styles.css

```
input.ng-invalid.ng-touched
{
  border: 2px solid red;
}
input.ng-valid.ng-touched
{
  border: 2px solid green;
}
.error
{
```

```
color: red;
}

c:\angular\app1\src\app\app.module.ts
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule, FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}

c:\angular\app1\src\app\app.component.ts
import { Component } from "@angular/core";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  firstname: string = null;
  lastname: string = null;
  email: string = null;
  amount: string = null;
  gender: string = null;
  country: string = "";
  msg: string = null;

  onRegisterClick(f)
  {
    if (f.valid)
    {
      this.msg = "Firstname: " + this.firstname + "<br>Lastname: " + this.lastname + "<br>Email: " +
      this.email + "<br>Amount: " + this.amount + "<br>Gender: " + this.gender + "<br>Country: " + this.country;
    }
    else
    {
      this.msg = "Invalid";
    }
  }
}
```

```

}

c:\angular\app1\src\app\app.component.html
<div class="class1">
  <h4>Template Drive Forms</h4>
  <form #myform="ngForm">
    Firstname:
    <input type="text" [(ngModel)]="firstname" name="username" required="required" minlength="3"
    maxlength="20" pattern="^[a-zA-Z ]*" #control1="ngModel">
    <span class="error" *ngIf="control1.touched && control1.invalid &&
    control1.errors.required">Firstname can't be blank</span>
    <span class="error" *ngIf="control1.touched && control1.invalid &&
    control1.errors.minlength">Min: 3 characters</span>
    <span class="error" *ngIf="control1.touched && control1.invalid &&
    control1.errors.pattern">Alphabets only allowed</span>
    <br>

    Lastname:
    <input type="text" [(ngModel)]="lastname" name="lastname" required="required" minlength="3"
    maxlength="20" pattern="^[a-zA-Z ]*" #control2="ngModel">
    <span class="error" *ngIf="control2.touched && control2.invalid &&
    control2.errors.required">Lastname can't be blank</span>
    <span class="error" *ngIf="control2.touched && control2.invalid &&
    control2.errors.minlength">Min: 3 characters</span>
    <span class="error" *ngIf="control2.touched && control2.invalid &&
    control2.errors.pattern">Alphabets only allowed</span>
    <br>

    Email:
    <input type="text" [(ngModel)]="email" name="email" required="required" pattern="^[a-zA-
    Z]+(\.[a-zA-Z0-9]+)*@[a-zA-Z-]+\(\.[a-zA-Z-]+\)*(\.[a-zA-Z]{2,15})$" #control3="ngModel">
    <span class="error" *ngIf="control3.touched && control3.invalid &&
    control3.errors.required">Email can't be blank</span>
    <span class="error" *ngIf="control3.touched && control3.invalid && control3.errors.pattern">Email
    is not valid</span>
    <br>

    Amount:
    <input type="text" [(ngModel)]="amount" name="amount" required="required" pattern="^([0-9]*$"
    #control4="ngModel">
    <span class="error" *ngIf="control4.touched && control4.invalid &&
    control4.errors.required">Amount can't be blank</span>
    <span class="error" *ngIf="control4.touched && control4.invalid &&
    control4.errors.pattern">Alphabets not allowed</span>
    <br>

    Gender:
    <input type="radio" [(ngModel)]="gender" value="male" name="gender" required="required"
    #control5="ngModel">
    Male
    <input type="radio" [(ngModel)]="gender" value="female" name="gender" required="required"
    #control5="ngModel">

```

Female
`Please Select Gender`
`
`

Country:
`<select [(ngModel)]="country" name="country" required="required" #control6="ngModel">`
`<option value="">Please Select</option>`
`<option>India</option>`
`<option>USA</option>`
`<option>UK</option>`
`<option>Japan</option>`
`</select>`
`Please Select Country`
`
`

`<input type="submit" value="Register" (click)="onRegisterClick(myform)">
`
`<div [innerHTML]="msg"></div>`
`</form>`
`</div>`

Executing the application:

- Open Command Prompt and enter the following commands:
`cd c:\angular\app1`
`ng serve`
- Open the browser and enter the following URL:
`http://localhost:4200`

Template Drive Forms

Firstname:

Lastname:

Email:

Amount:

Gender: Male Female

Country:

Firstname: Adam
Lastname: Smith
Email: adam@gmail.com
Amount: 2000
Gender: male
Country: India

Reactive Forms (or) Model Driven Forms

- Reactive Forms (or) Model Driven Forms are new types of forms in angular, which are suitable for creating large forms with many fields and complex validations.
- In these forms, each field is represented as “FormControl” and group of controls is represented as “FormGroup”.
- “ReactiveFormsModule” should be imported from “@angular/forms” package.
- Validation rules are defined in the component using "Validators" object of angular and validation messages are displayed in the template using "validation properties" of angular.

Validations in Reactive Forms:

- Validators.required : Field is mandatory
- Validators.minLength : Minimum no. of characters
- Validators.maxLength : Maximum no. of characters
- Validators.pattern : Regular expression

Validation Properties:

- untouched
 - true : Field is not focused.
 - false : Field is focused.
- touched
 - true : Field is focused.
 - false : Field is not focused.
- pristine
 - true : Field is not modified by the user.
 - false : Field is modified by the user.
- dirty
 - true : Field is modified by the user.
 - false : Field is not modified by the user.
- valid
 - true : Field value is valid.
 - false : Field value is invalid
- invalid
 - true : Field value is invalid.
 - false : Field value is valid.
- errors : Represents the list of errors of the field.
 - required : true / false
 - minlength : true / false
 - maxlength : true / false
 - pattern : true / false

Reactive Forms - Example

- We are going to create a sample reactive form with validations.

Fields:

- Firstname
- Lastname
- Email
- Amount
- Gender
- Country

Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular  
ng new app1  
cd c:\angular\app1
```

c:\angular\app1\package.json

```
{  
  "name": "app1",  
  "version": "0.0.0",  
  "license": "MIT",  
  "scripts": {  
    "ng": "ng",  
    "start": "ng serve",  
    "build": "ng build --prod",  
    "test": "ng test",  
    "lint": "ng lint",  
    "e2e": "ng e2e"  
  },  
  "private": true,  
  "dependencies": {  
    "@angular/animations": "^5.2.0",  
    "@angular/common": "^5.2.0",  
    "@angular/compiler": "^5.2.0",  
    "@angular/core": "^5.2.0",  
    "@angular/forms": "^5.2.0",  
    "@angular/http": "^5.2.0",  
    "@angular/platform-browser": "^5.2.0",  
    "@angular/platform-browser-dynamic": "^5.2.0",  
    "@angular/router": "^5.2.0",  
    "core-js": "^2.4.1",  
    "rxjs": "^5.5.6",  
    "zone.js": "^0.8.19"  
  },  
  "devDependencies": {  
    "@angular/cli": "~1.7.4",  
    "@angular/compiler-cli": "^5.2.0",  
    "@angular/language-service": "^5.2.0",  
    "@types/jasmine": "~2.8.3",  
    "@types/jasminewd2": "~2.0.2",  
    "@types/node": "~6.0.60",  
    "codelyzer": "^4.0.1",  
    "jasmine-core": "~2.8.0",  
    "jasmine-spec-reporter": "~4.2.1",  
    "karma": "~2.0.0",  
    "karma-chrome-launcher": "~2.2.0",  
    "karma-coverage-istanbul-reporter": "^1.2.1",  
    "karma-jasmine": "~1.1.0",  
    "karma-jasmine-html-reporter": "^0.2.2",  
    "protractor": "~5.1.2",  
  }  
}
```

```
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\styles.css

```
input.ng-invalid.ng-touched
{
    border: 2px solid red;
}
input.ng-valid.ng-touched
{
    border: 2px solid green;
}
.error
{
    color: red;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { ReactiveFormsModule } from "@angular/forms";

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule, ReactiveFormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";
import { FormGroup, FormControl, Validators } from "@angular/forms";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{
  msg: string = "";
  myform: FormGroup;
```

```

constructor()
{
  this.myform = new FormGroup({
    firstname: new FormControl("", [Validators.required, Validators.minLength(3),
    Validators.maxLength(20), Validators.pattern("^[a-zA-Z ]*$")]),
    lastname: new FormControl("", [Validators.required, Validators.minLength(5),
    Validators.maxLength(20), Validators.pattern("^[a-zA-Z ]*$")]),
    email: new FormControl("", [Validators.required, Validators.pattern("^[a-z0-9]+(\._[a-z0-9]+)*@[a-z0-
    9-]+\([a-z0-9-]+\)([a-z]{2,15})$")]),
    amount: new FormControl("", [Validators.required, Validators.pattern("^[0-9]*$")]),
    gender: new FormControl("", [Validators.required]),
    country: new FormControl("", [Validators.required])
  });
}

onRegisterClick()
{
  if(this.myform.valid)
  {
    this.msg = "First Name: " + this.myform.controls.firstname.value + "<br>Last Name: " +
    this.myform.controls.lastname.value + "<br>Email: " + this.myform.controls.email.value + "<br>Amount: " +
    + this.myform.controls.amount.value + "<br>Gender: " + this.myform.controls.gender.value +
    "<br>Country: " + this.myform.controls.country.value;
  }
  else
  {
    this.msg = "Invalid";
  }
}
}

```

c:\angular\app1\src\app\app.component.html

```

<div class="class1">
  <h4>Reactive Drive Forms</h4>

  <form [formGroup]="myform">
    First Name:
    <input type="text" formControlName="firstname">
    <span class="error" *ngIf="myform.controls.firstname.touched && myform.controls.firstname.invalid
    && myform.controls.firstname.errors.required">Firstname can't be blank</span>
    <span class="error" *ngIf="myform.controls.firstname.touched && myform.controls.firstname.invalid
    && myform.controls.firstname.errors.minLength">Min: 5 characters</span>
    <span class="error" *ngIf="myform.controls.firstname.touched && myform.controls.firstname.invalid
    && myform.controls.firstname.errors.maxLength">Max: 20 characters</span>
    <span class="error" *ngIf="myform.controls.firstname.touched && myform.controls.firstname.invalid
    && myform.controls.firstname.errors.pattern">Alphabets only allowed</span>
    <br>

    Last Name:
    <input type="text" formControlName="lastname">
  
```

```
<span class="error" *ngIf="myform.controls.lastname.touched && myform.controls.lastname.invalid && myform.controls.lastname.errors.required">Lastname can't be blank</span>
<span class="error" *ngIf="myform.controls.lastname.touched && myform.controls.lastname.invalid && myform.controls.lastname.errors.minLength">Min: 5 characters</span>
<span class="error" *ngIf="myform.controls.lastname.touched && myform.controls.lastname.invalid && myform.controls.lastname.errors.maxLength">Max: 20 characters</span>
<span class="error" *ngIf="myform.controls.lastname.touched && myform.controls.lastname.invalid && myform.controls.lastname.errors.pattern">Alphabets only allowed</span>
<br>
```

Email:

```
<input type="text" formControlName="email">
<span class="error" *ngIf="myform.controls.email.touched && myform.controls.email.invalid && myform.controls.email.errors.required">Email can't be blank</span>
<span class="error" *ngIf="myform.controls.email.touched && myform.controls.email.invalid && myform.controls.email.errors.pattern">Email is not valid</span>
<br>
```

Amount:

```
<input type="text" formControlName="amount">
<span class="error" *ngIf="myform.controls.amount.touched && myform.controls.amount.invalid && myform.controls.amount.errors.required">Amount can't be blank</span>
<span class="error" *ngIf="myform.controls.amount.touched && myform.controls.amount.invalid && myform.controls.amount.errors.pattern">Alphabets not allowed</span>
<br>
```

Gender:

```
<input type="radio" formControlName="gender" value="male">
Male
<input type="radio" formControlName="gender" value="female">
Female
<span class="error" *ngIf="myform.controls.gender.touched && myform.controls.gender.invalid && myform.controls.gender.errors.required">Please Select Gender</span>
<br>
```

Country:

```
<select formControlName="country">
<option>Please Select</option>
<option>India</option>
<option>USA</option>
<option>UK</option>
<option>Japan</option>
</select>
<span class="error" *ngIf="myform.controls.country.touched && myform.controls.country.invalid && myform.controls.country.errors.required">Please Select Country</span>
<br>

<input type="submit" value="Submit" (click)="onRegisterClick()"><br>
<div [innerHTML]="msg"></div>
</form>
</div>
```

Executing the application:

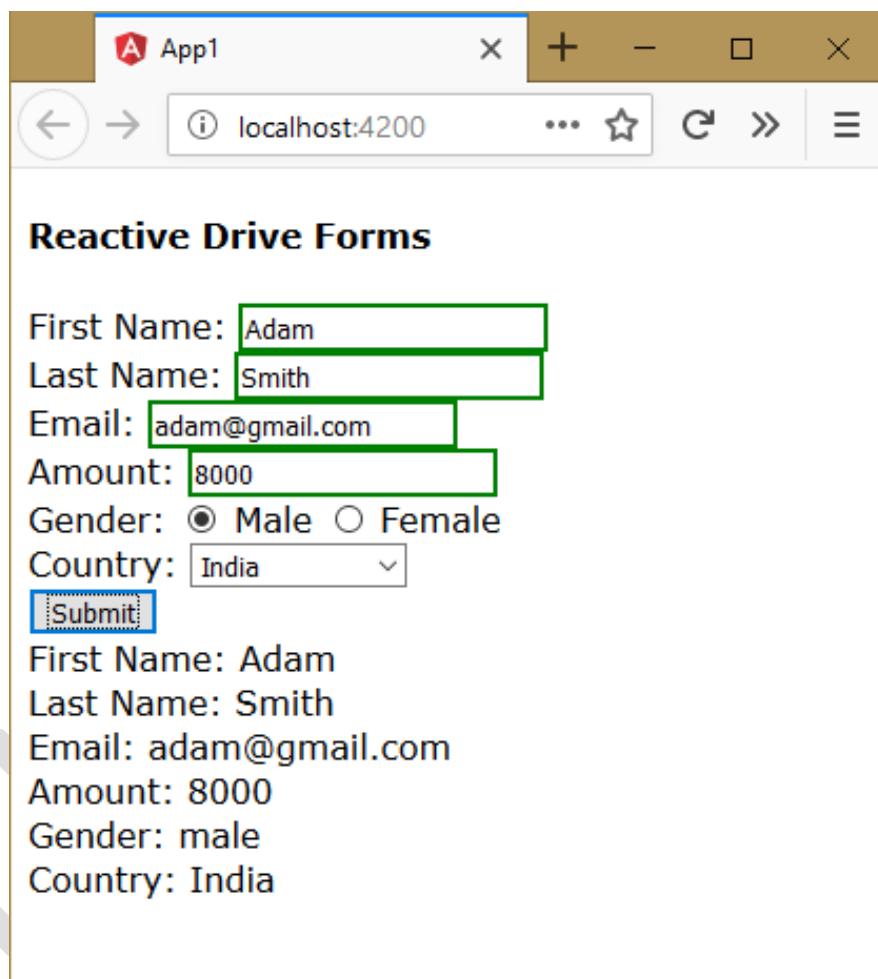
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



Routing

- The “Routing” concept is used to create page navigation in angular 2+ applications.
- “Routing” includes the process of mapping between the “route (url)” and corresponding component. Ex:
 - `http://localhost:8080/home` → HomeComponent
 - `http://localhost:8080/about` → AboutComponent
- The “`@angular/router`” package provides essential API to create routing.
- Angular 2+ supports two types of routing.
 1. Hash-less routing Ex: /home

2. Hash routing Ex: #/home

Steps for working with Routing

- Import “@angular/router” package in “package.json” file:

```
“dependencies”:  
{  
    “@angular/router”: “latest”  
}
```

- Set the base location of the application on server:

```
<base href="/">
```

- Import “Router” from “@angular/router” package:

```
Import { Routes } from “@angular/router”;
```

- Create routes:

```
var variable1 : Routes = [  
    { path: “path here”, component: ComponentClassName },  
    { path: “path here”, component: ComponentClassName },  
    ....  
];
```

- Import “RouterModule” from “@angular/router” package:

```
Import { RouterModule} from “@angular/router”;
```

- Combine “your routes” and “RouterModule”:

```
var variable2 = RouterModule.forRoot(variable1, { useHash: true/false } );
```

- Import both “routes” and “RouterModule” in “AppModule”:

```
@NgModule( { ..., imports: [ ..., variable2 ] } )  
class AppModule()  
{  
}
```

- Create hyperlink to route:

```
<a routerLink="/path">Link text</a>
```

- Create placeholder to display route content:

```
<router-outlet>  
</router-outlet>
```

Routing - Example

Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g component Home
ng g component About
ng g component Contact
```

c:\angular\app1\package.json

```
{
  "name": "app1",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build --prod",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^5.2.0",
    "@angular/common": "^5.2.0",
    "@angular/compiler": "^5.2.0",
    "@angular/core": "^5.2.0",
    "@angular/forms": "^5.2.0",
    "@angular/http": "^5.2.0",
    "@angular/platform-browser": "^5.2.0",
    "@angular/platform-browser-dynamic": "^5.2.0",
    "@angular/router": "^5.2.0",
    "core-js": "^2.4.1",
    "rxjs": "^5.5.6",
    "zone.js": "^0.8.19"
  },
  "devDependencies": {
    "@angular/cli": "~1.7.4",
    "@angular/compiler-cli": "^5.2.0",
    "@angular/language-service": "^5.2.0",
    "@types/jasmine": "~2.8.3",
    "@types/jasminewd2": "~2.0.2",
    "@types/node": "~6.0.60",
    "codelyzer": "^4.0.1",
  }
}
```

```
"jasmine-core": "~2.8.0",
"jasmine-spec-reporter": "~4.2.1",
"karma": "~2.0.0",
"karma-chrome-launcher": "~2.2.0",
"karma-coverage-istanbul-reporter": "^1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\styles.css

```
#container
{
    background-color: #ccccff;
    margin: 5px;
    padding: 5px;
    border-radius: 5px;
    height: 200px;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { ContactComponent } from './contact/contact.component';

var myroutes: Routes = [
    { path: "", component: HomeComponent },
    { path: "home", component: HomeComponent },
    { path: "about", component: AboutComponent },
    { path: "contact", component: ContactComponent }
];
var myroutes2 = RouterModule.forRoot(myroutes);

@NgModule({
    declarations: [
        AppComponent,
        HomeComponent,
        AboutComponent,
        ContactComponent
    ],
    imports: [
        BrowserModule, FormsModule, myroutes2
]
```

```
],
providers: [],
bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div class="class1">
  <h4>Routing</h4>
  <a routerLink="home">Home</a>
  <a routerLink="about">About</a>
  <a routerLink="contact">Contact</a>

  <div id="container">
    <router-outlet>
    </router-outlet>
  </div>
</div>
```

c:\angular\app1\src\app\home\home.component.ts

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

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {

  constructor() {}

  ngOnInit() {
  }
}
```

c:\angular\app1\src\app\home\home.component.html

```
<div class="class1">
  <h5>Home</h5>
```

```
</div>
```

c:\angular\app1\src\app\about\about.component.ts

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

@Component({
  selector: 'app-about',
  templateUrl: './about.component.html',
  styleUrls: ['./about.component.css']
})
export class AboutComponent implements OnInit {

  constructor() {}

  ngOnInit() {
  }
}
```

c:\angular\app1\src\app\about\about.component.html

```
<div class="class1">
<h5>About</h5>
</div>
```

c:\angular\app1\src\app\contact\contact.component.ts

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

@Component({
  selector: 'app-contact',
  templateUrl: './contact.component.html',
  styleUrls: ['./contact.component.css']
})
export class ContactComponent implements OnInit {

  constructor() {}

  ngOnInit() {
  }
}
```

c:\angular\app1\src\app\contact\contact.component.html

```
<div class="class1">
<h5>Contact</h5>
</div>
```

Executing the application:

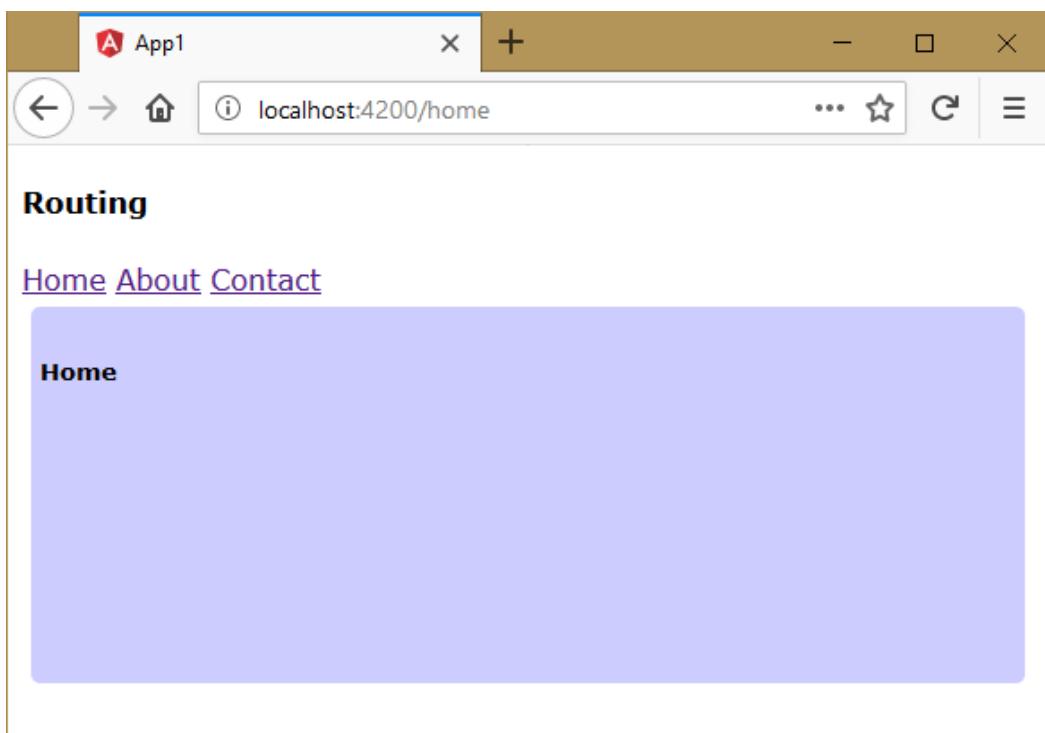
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

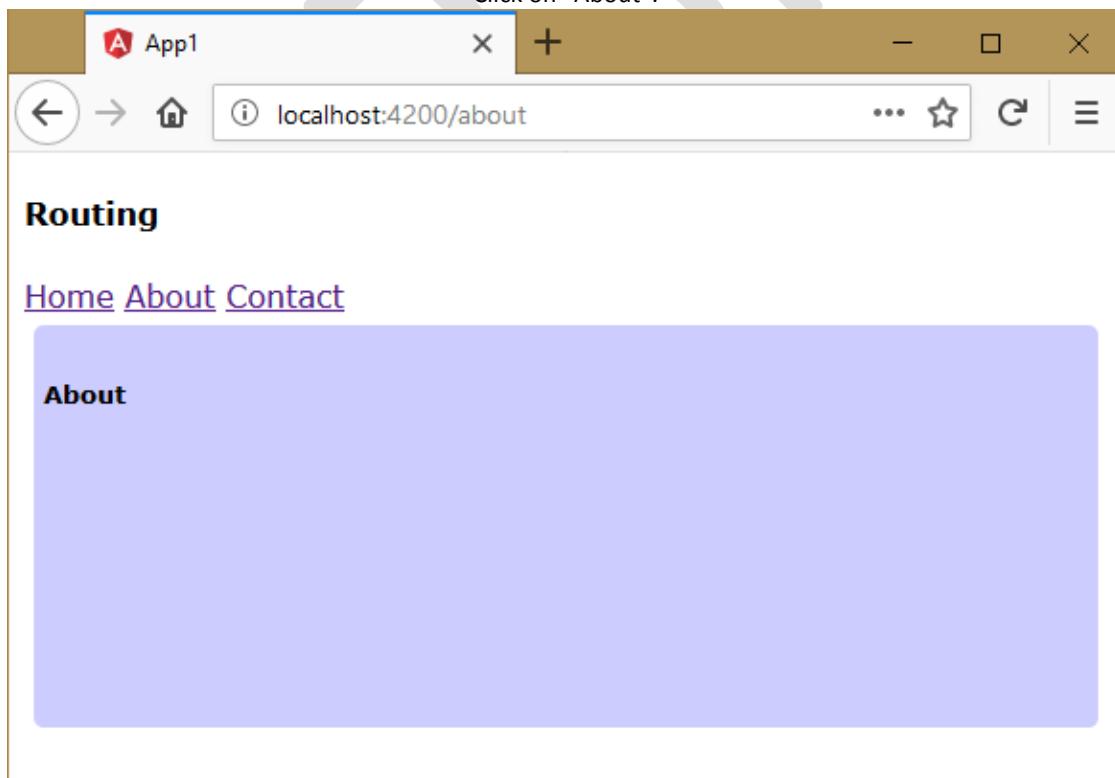
```
ng serve
```

- Open the browser and enter the following URL:

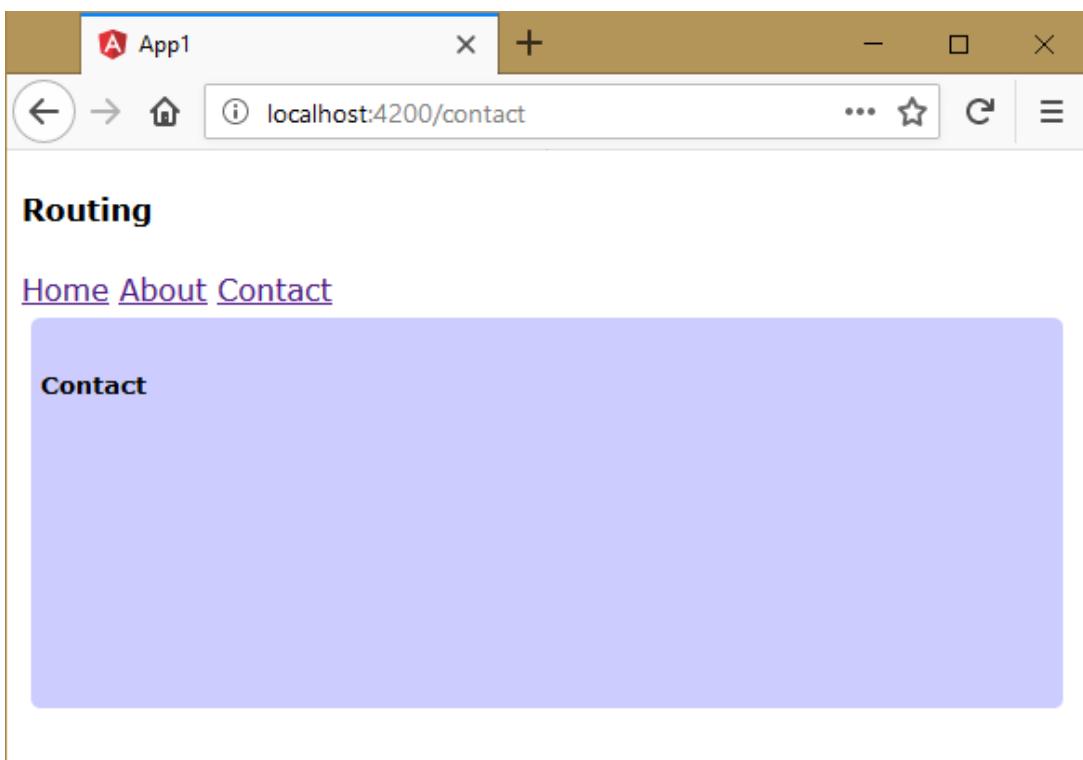
http://localhost:4200



Click on "About".



Click on "Contact".



Route Parameters

- You can pass parameters to the route.
- Route parameter is represented as “:parametername” syntax.
- You can get the value of the parameter in the component using “ActivatedRoute” service.

Steps for Working with Route Parameters

- Create parameter in the route:

```
{ path: "pathname/:parametername", component: ComponentClassname }
```

- Import the "ActivatedRoute" service:

```
import { ActivatedRoute } from "@angular/router";
```

- Get an object of "ActivatedRoute" service:

```
constructor(@Inject(ActivatedRoute) private route : ActivatedRoute)
{
}
```

- Get the value of parameter:

```
this.route.snapshot.params["parametername"]
```

- (or) Get the value of parameter with updates:

```
this.route.params.subscribe(params =>  
{  
  params["parametername"]  
});
```

Route Parameters - Example

Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular  
ng new app1  
cd c:\angular\app1  
ng g component Home  
ng g component About  
ng g component Contact  
ng g component Products  
ng g class Product  
ng g service Products
```

c:\angular\app1\package.json

```
{  
  "name": "app1",  
  "version": "0.0.0",  
  "license": "MIT",  
  "scripts": {  
    "ng": "ng",  
    "start": "ng serve",  
    "build": "ng build --prod",  
    "test": "ng test",  
    "lint": "ng lint",  
    "e2e": "ng e2e"  
},  
  "private": true,  
  "dependencies": {  
    "@angular/animations": "^5.2.0",  
    "@angular/common": "^5.2.0",  

```

```
"@angular/platform-browser-dynamic": "^5.2.0",
"@angular/router": "^5.2.0",
"core-js": "^2.4.1",
"rxjs": "^5.5.6",
"zone.js": "^0.8.19"
},
"devDependencies":{
"@angular/cli": "~1.7.4",
"@angular/compiler-cli": "^5.2.0",
"@angular/language-service": "^5.2.0",
"@types/jasmine": "~2.8.3",
"@types/jasminewd2": "~2.0.2",
"@types/node": "~6.0.60",
"codelyzer": "^4.0.1",
"jasmine-core": "~2.8.0",
"jasmine-spec-reporter": "~4.2.1",
"karma": "~2.0.0",
"karma-chrome-launcher": "~2.2.0",
"karma-coverage-istanbul-reporter": "^1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\styles.css

```
#container
{
  background-color: #ccccff;
  margin: 5px;
  padding: 5px;
  border-radius: 5px;
  height: 200px;
}
```

c:\angular\app1\src\product.ts

```
export class Product
{
  productId: number;
  productName: string;
  cost: number;
  brand: string;

  constructor(productId: number, productName: string, cost: number, brand: string)
  {
    this.productId = productId;
    this.productName = productName;
    this.cost = cost;
```

```
this.brand = brand;
}
}

c:\angular\app1\src\products.service.ts

import { Injectable } from '@angular/core';
import { Product } from './product';

@Injectable()
export class ProductsService
{
  products: Product[];

  constructor()
  {
    this.products = [
      new Product(101, "Samsung S8", 40000, "Samsung"),
      new Product(101, "Samsung S9", 62000, "Samsung"),
      new Product(101, "iPhone 8", 60000, "Apple"),
      new Product(101, "iPhone 10", 98000, "Apple"),
      new Product(101, "Pixel 2", 53000, "Google"),
      new Product(101, "Pixel 3", 95000, "Google"),
    ];
  }
}
```

```
getProductsByBrand(brandName: string): Product[]
{
  var selectedProducts : Product[] = [];
  for (var i = 0; i < this.products.length; i++)
  {
    if (this.products[i].brand == brandName)
    {
      selectedProducts.push(this.products[i]);
    }
  }

  return selectedProducts;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { ContactComponent } from './contact/contact.component';
import { ProductsComponent } from './products/products.component';
import { ProductsService } from './products.service';
```

```
var myroutes: Routes = [
  { path: "", component: HomeComponent },
  { path: "home", component: HomeComponent },
  { path: "about", component: AboutComponent },
  { path: "contact", component: ContactComponent },
  { path: "products/:brandname", component: ProductsComponent }
];
var myroutes2 = RouterModule.forRoot(myroutes);

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    AboutComponent,
    ContactComponent,
    ProductsComponent
  ],
  imports: [
    BrowserModule, FormsModule, myroutes2
  ],
  providers: [ ProductService ],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div class="class1">
  <h4>Routing</h4>
  <a routerLink="home">Home</a>
  <a routerLink="about">About</a>
  <a routerLink="contact">Contact</a>
  <a routerLink="products/Samsung">Samsung</a>
  <a routerLink="products/Apple">Apple</a>
  <a routerLink="products/Google">Google</a>

  <div id="container">
    <router-outlet>
    </router-outlet>
```

```
</div>
</div>
```

c:\angular\app1\src\app\home\home.component.ts

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

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {

  constructor() {}

  ngOnInit() {
  }

}
```

c:\angular\app1\src\app\home\home.component.html

```
<div class="class1">
  <h5>Home</h5>
</div>
```

c:\angular\app1\src\app\about\about.component.ts

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

@Component({
  selector: 'app-about',
  templateUrl: './about.component.html',
  styleUrls: ['./about.component.css']
})
export class AboutComponent implements OnInit {

  constructor() {}

  ngOnInit() {
  }

}
```

c:\angular\app1\src\app\about\about.component.html

```
<div class="class1">
  <h5>About</h5>
</div>
```

c:\angular\app1\src\app\contact\contact.component.ts

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

@Component({
```

```
    selector: 'app-contact',
    templateUrl: './contact.component.html',
    styleUrls: ['./contact.component.css']
  })
export class ContactComponent implements OnInit {

  constructor() {}

  ngOnInit() {
  }
}
```

c:\angular\app1\src\app\contact\contact.component.html

```
<div class="class1">
  <h5>Contact</h5>
</div>
```

c:\angular\app1\src\app\products\products.component.ts

```
import { Component, OnInit, Inject } from '@angular/core';
import { Product } from './product';
import { ProductsService } from './products.service';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-products',
  templateUrl: './products.component.html',
  styleUrls: ['./products.component.css']
})
export class ProductsComponent implements OnInit
{
  brand: string;
  matchingproducts: Product[] = [];

  constructor(@Inject(ProductsService) private prodService : ProductsService, @Inject(ActivatedRoute) private route: ActivatedRoute)
  {
  }

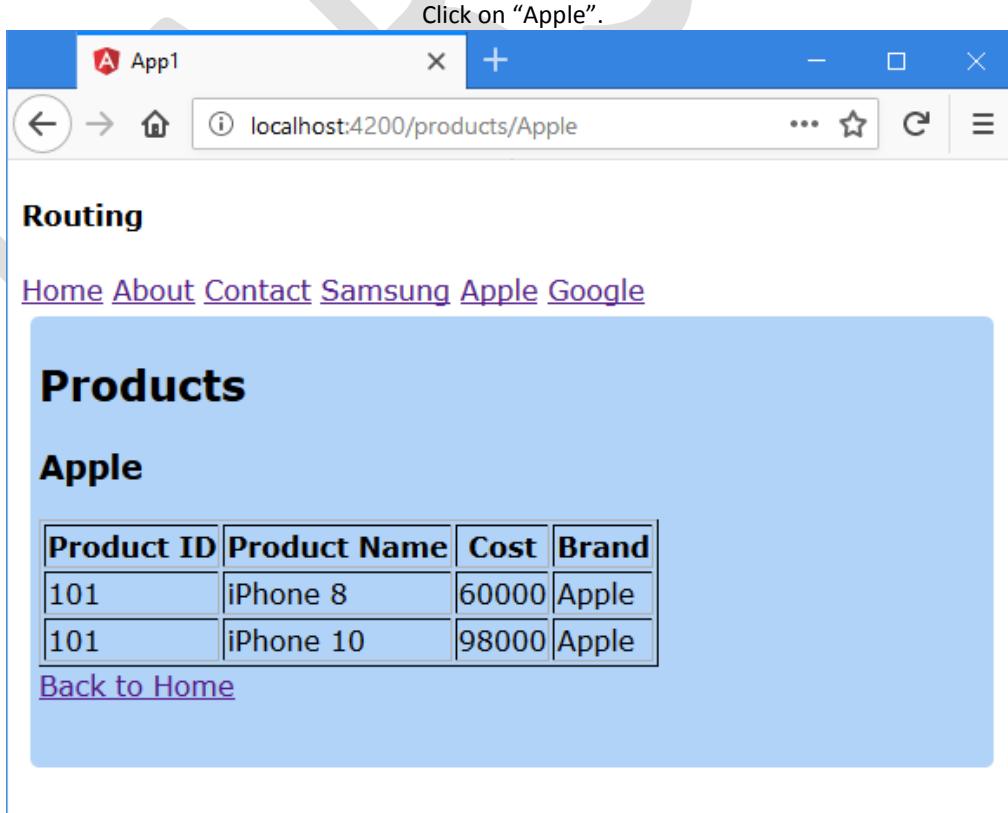
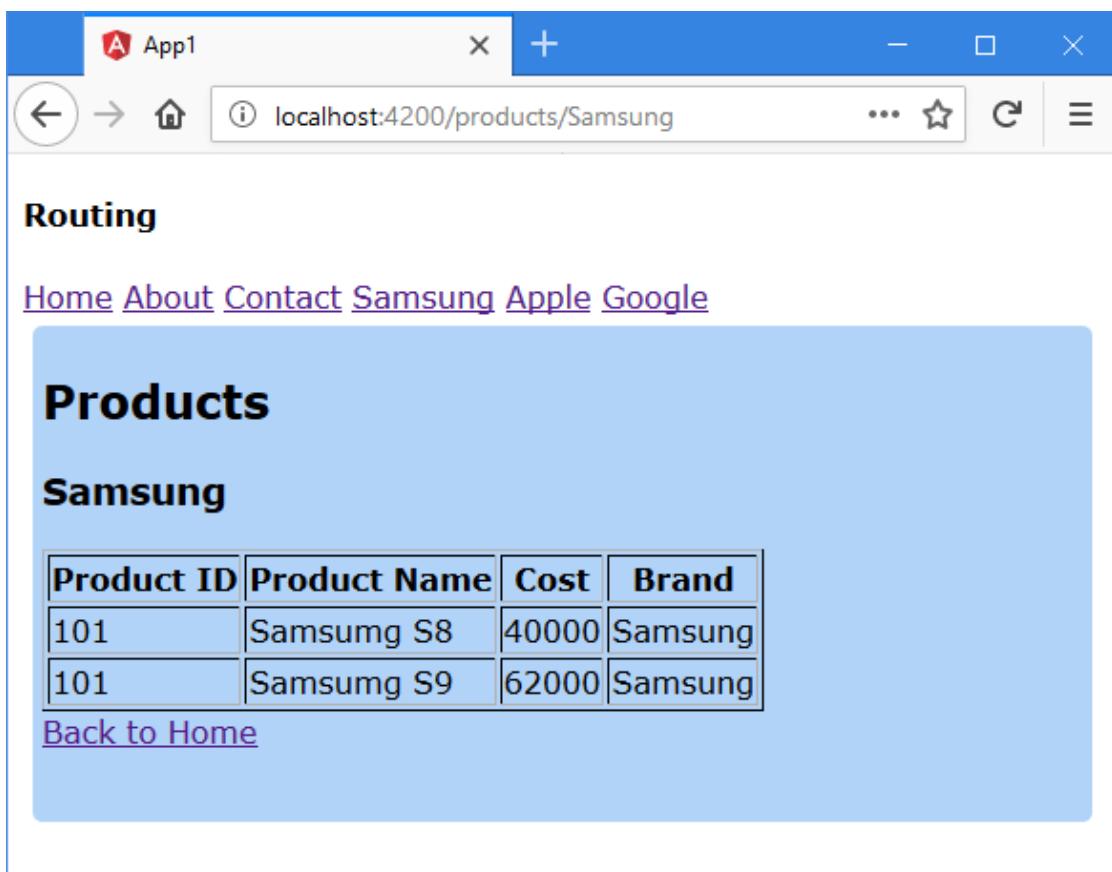
  ngOnInit()
  {
    this.route.params.subscribe(params =>
    {
      var selectedBrand = params["brandname"];
      this.brand = selectedBrand;
      this.matchingproducts = this.prodService.getProductsByBrand(selectedBrand);
    });
  }
}
```

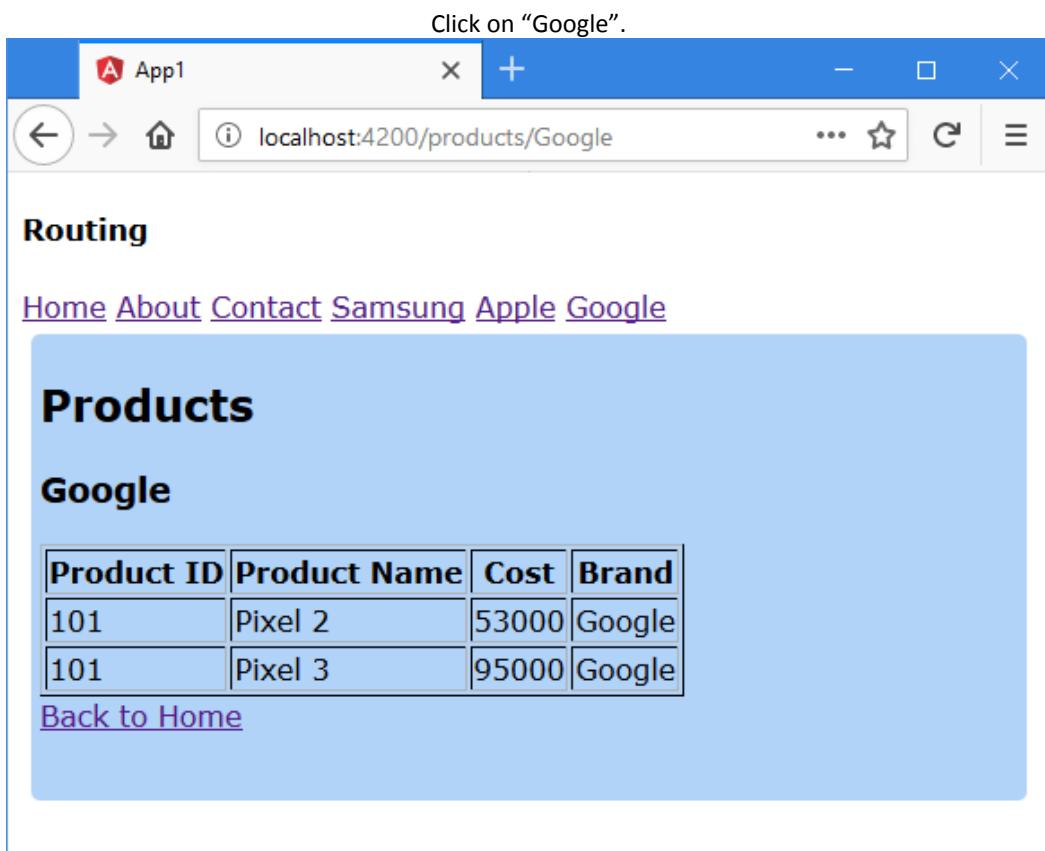
c:\angular\app1\src\app\products\products.component.html

```
<div>
  <h2>Products</h2>
  <h3>{{brand}}</h3>
  <table border="1">
    <tr>
      <th>Product ID</th>
      <th>Product Name</th>
      <th>Cost</th>
      <th>Brand</th>
    </tr>
    <tr *ngFor="let product of matchingproducts">
      <td>{{product.productId}}</td>
      <td>{{product.productName}}</td>
      <td>{{product.cost}}</td>
      <td>{{product.brand}}</td>
    </tr>
  </table>
  <a routerLink="/home">Back to Home</a>
</div>
```

Executing the application:

- Open Command Prompt and enter the following commands:
cd c:\angular\app1
ng serve
- Open the browser and enter the following URL:
<http://localhost:4200>





Child Routes

- Route can have child routes up to unlimited no. of nested levels.
- Ex: "Funds Transfer" menu has "Transfer", "Add Payee", "Activate Payee" etc.

Steps for Working with Child Routes

- **Create Child Routes:**

```
{ path: "parentpath", component: ComponentClassname, children: [
  { path: "childpath", component: ComponentClassname },
  { path: "childpath", component: ComponentClassname },
  ...
]}
```

- **Create hyperlink for the child route:**

```
<a href="/parentpath/childpath">Link text</a>
```

- **Create router outlet for child routes (in the parent route component's template):**

```
<router-outlet></router-outlet>
```

Child Routes - Example

Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g component OnlineShopping
ng g component Appliances
ng g component Electronics
ng g component Fashion
ng g component Furniture
ng g component Lighting
ng g component Mobiles
ng g component Laptops
ng g component Men
ng g component Women
ng g component Furniture
```

c:\angular\app1\package.json

```
{
  "name": "app1",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build --prod",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^5.2.0",
    "@angular/common": "^5.2.0",
    "@angular/compiler": "^5.2.0",
    "@angular/core": "^5.2.0",
    "@angular/forms": "^5.2.0",
    "@angular/http": "^5.2.0",
    "@angular/platform-browser": "^5.2.0",
    "@angular/platform-browser-dynamic": "^5.2.0",
    "core-js": "2.4.1"
  }
}
```

```
"@angular/router": "^5.2.0",
"core-js": "^2.4.1",
"rxjs": "^5.5.6",
"zone.js": "^0.8.19"
},
"devDependencies": {
"@angular/cli": "~1.7.4",
"@angular/compiler-cli": "^5.2.0",
"@angular/language-service": "^5.2.0",
"@types/jasmine": "~2.8.3",
"@types/jasminewd2": "~2.0.2",
"@types/node": "~6.0.60",
"codemlyzer": "^4.0.1",
"jasmine-core": "~2.8.0",
"jasmine-spec-reporter": "~4.2.1",
"karma": "~2.0.0",
"karma-chrome-launcher": "~2.2.0",
"karma-coverage-istanbul-reporter": "^1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\styles.css

```
#container
{
  background-color: #b2d3f8;
  margin: 5px;
  padding: 5px;
  border-radius: 5px;
  height: 350px;
}

.container2
{
  background-color: #d8f696;
  margin: 5px;
  padding: 5px;
  border-radius: 3px;
  height: 150px;
}
```

c:\angular\app1\src\app\app.module.ts

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

```
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";
import { Routes, RouterModule } from "@angular/router";
import { AppliancesComponent } from './appliances/appliances.component';
import { ElectronicsComponent } from './electronics/electronics.component';
import { FashionComponent } from './fashion/fashion.component';
import { FurnitureComponent } from './furniture/furniture.component';
import { LaptopsComponent } from './laptops/laptops.component';
import { LightingComponent } from './lighting/lighting.component';
import { MenComponent } from './men/men.component';
import { MobilesComponent } from './mobiles/mobiles.component';
import { OnlineShoppingComponent } from './online-shopping/online-shopping.component';
import { WomenComponent } from './women/women.component';

var myroutes: Routes = [
  { path: "", component: OnlineShoppingComponent },
  {
    path: "electronics", component: ElectronicsComponent, children: [
      { path: "mobiles", component: MobilesComponent },
      { path: "laptops", component: LaptopsComponent },
    ]
  },
  {
    path: "appliances", component: AppliancesComponent, children: [
      { path: "lighting", component: LightingComponent },
      { path: "furniture", component: FurnitureComponent },
    ]
  },
  {
    path: "fashion", component: FashionComponent, children: [
      { path: "men", component: MenComponent },
      { path: "women", component: WomenComponent },
    ]
  }
];
var myroutes2 = RouterModule.forRoot(myroutes);

@NgModule({
  declarations: [
    AppComponent,
    AppliancesComponent,
    ElectronicsComponent,
    FashionComponent,
    FurnitureComponent,
    LaptopsComponent,
    LightingComponent,
    MenComponent,
    MobilesComponent,
    OnlineShoppingComponent,
    WomenComponent
  ],
})
```

```
imports: [
  BrowserModule, FormsModule, myroutes2
],
providers: [ ],
bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div class="class1">
  <h4>Child Routes</h4>
  <div id="container">
    <router-outlet>
      </router-outlet>
    </div>
  </div>
```

c:\angular\app1\src\app\online-shopping\online-shopping.component.ts

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

@Component({
  selector: 'app-online-shopping',
  templateUrl: './online-shopping.component.html',
  styleUrls: ['./online-shopping.component.css']
})
export class OnlineShoppingComponent implements OnInit {

  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\online-shopping\online-shopping.component.html

```
<div>
  <h4>Online Shopping</h4>
  <a routerLink="electronics">Electronics</a><br>
  <a routerLink="appliances">Appliances</a><br>
```

```
<a routerLink="fashion">Fashion</a><br>
</div>
```

c:\angular\app1\src\app\appliances\appliances.component.ts

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

@Component({
  selector: 'app-appliances',
  templateUrl: './appliances.component.html',
  styleUrls: ['./appliances.component.css']
})
export class AppliancesComponent implements OnInit {

  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\appliances\appliances.component.html

```
<div>
  <h4>Appliances</h4>
  <a routerLink="lighting">Lighting</a><br>
  <a routerLink="furniture">Furniture</a><br>
  <a routerLink="/">Home</a><br>
  <div class="container2">
    <router-outlet>
    </router-outlet>
  </div>
</div>
```

c:\angular\app1\src\app\electronics\electronics.component.ts

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

@Component({
  selector: 'app-electronics',
  templateUrl: './electronics.component.html',
  styleUrls: ['./electronics.component.css']
})
export class ElectronicsComponent implements OnInit {

  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\electronics\electronics.component.html

```
<div>
```

```
<h4>Electronics</h4>
<a routerLink="mobiles">Mobiles</a><br>
<a routerLink="laptops">Laptops</a><br>
<a routerLink="/">Home</a><br>
<div class="container2">
  <router-outlet>
  </router-outlet>
</div>
</div>
```

c:\angular\app1\src\app\fashion\fashion.component.ts

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

@Component({
  selector: 'app-fashion',
  templateUrl: './fashion.component.html',
  styleUrls: ['./fashion.component.css']
})
export class FashionComponent implements OnInit {

  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\fashion\fashion.component.html

```
<div>
  <h4>Fashion</h4>
  <a routerLink="men">Men</a><br>
  <a routerLink="women">Women</a><br>
  <a routerLink="/">Home</a><br>
  <div class="container2">
    <router-outlet>
    </router-outlet>
  </div>
</div>
```

c:\angular\app1\src\app\furniture\furniture.component.ts

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

@Component({
  selector: 'app-furniture',
  templateUrl: './furniture.component.html',
  styleUrls: ['./furniture.component.css']
})
export class FurnitureComponent implements OnInit {

  constructor() {}
```

```
ngOnInit()
{
}
}
```

c:\angular\app1\src\app\furniture\furniture.component.html

```
<div>
<h5>Furniture</h5>
</div>
```

c:\angular\app1\src\app\lighting\lighting.component.ts

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

@Component({
  selector: 'app-lighting',
  templateUrl: './lighting.component.html',
  styleUrls: ['./lighting.component.css']
})
export class LightingComponent implements OnInit {

  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\lighting\lighting.component.html

```
<div>
<h5>Lighting</h5>
</div>
```

c:\angular\app1\src\app\mobiles\mobiles.component.ts

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

@Component({
  selector: 'app-mobiles',
  templateUrl: './mobiles.component.html',
  styleUrls: ['./mobiles.component.css']
})
export class MobilesComponent implements OnInit {

  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\mobiles\mobiles.component.html

```
<div>
  <h5>Mobiles</h5>
</div>
```

c:\angular\app1\src\app\laptops\laptops.component.ts

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

@Component({
  selector: 'app-laptops',
  templateUrl: './laptops.component.html',
  styleUrls: ['./laptops.component.css']
})
export class LaptopsComponent implements OnInit {

  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\laptops\laptops.component.html

```
<div>
  <h5>Laptops</h5>
</div>
```

c:\angular\app1\src\app\men\men.component.ts

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

@Component({
  selector: 'app-men',
  templateUrl: './men.component.html',
  styleUrls: ['./men.component.css']
})
export class MenComponent implements OnInit
{
  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\men\men.component.html

```
<div>
  <h5>Men</h5>
</div>
```

c:\angular\app1\src\app\women\women.component.ts

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

@Component({
  selector: 'app-women',
  templateUrl: './women.component.html',
  styleUrls: ['./women.component.css']
})
export class WomenComponent implements OnInit
{

  constructor() {}

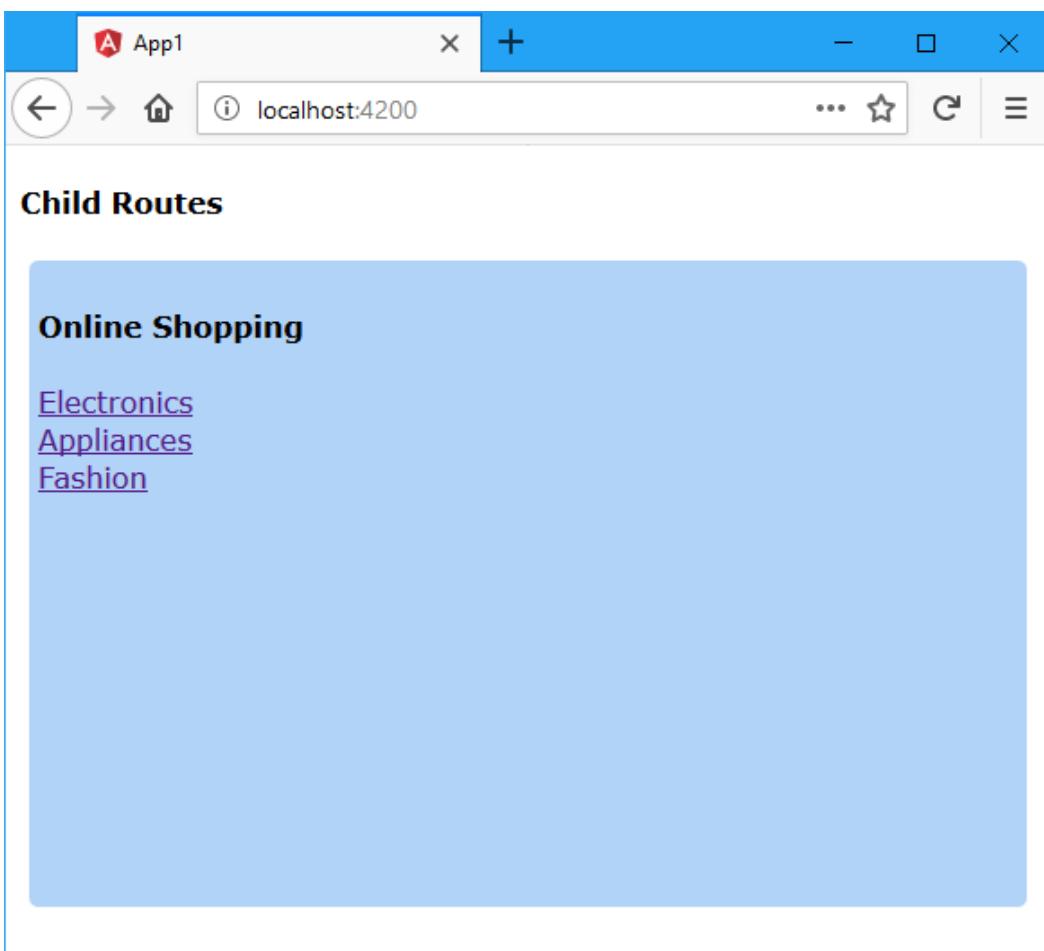
  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\women\women.component.html

```
<div>
<h5>Women</h5>
</div>
```

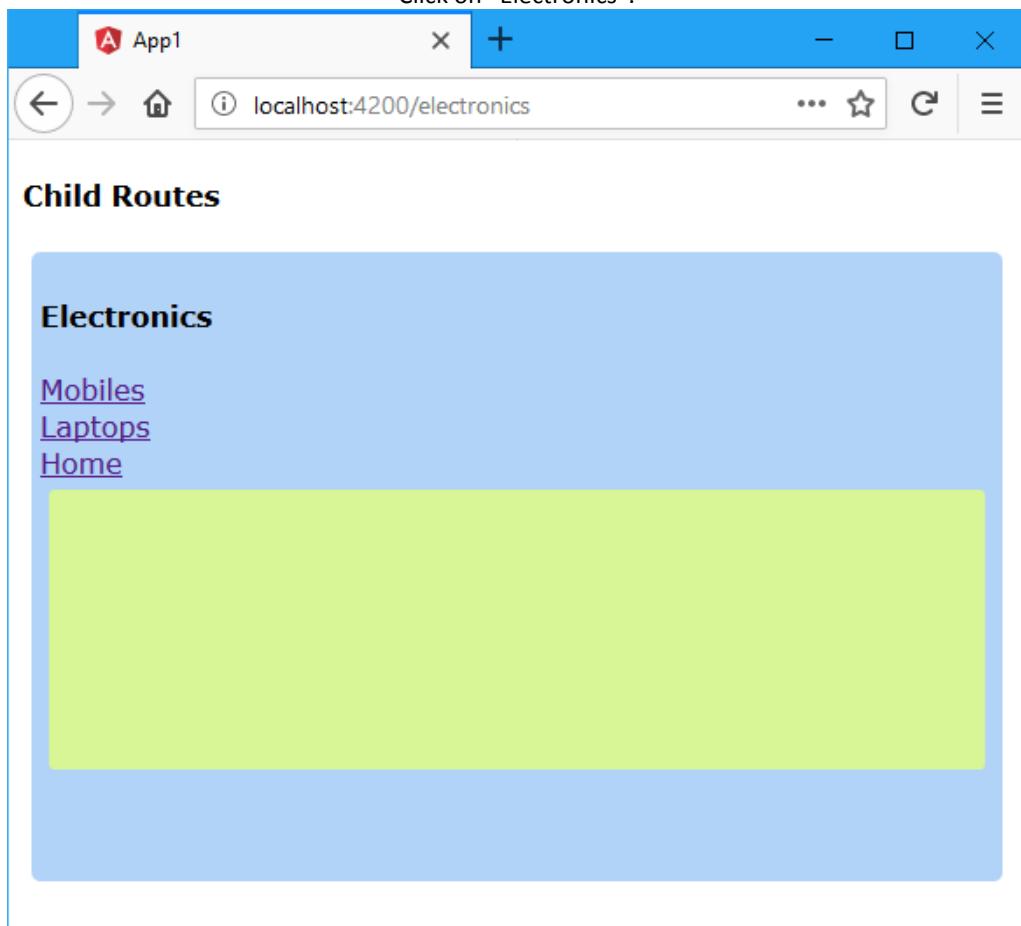
Executing the application:

- Open Command Prompt and enter the following commands:
cd c:\angular\app1
ng serve
- Open the browser and enter the following URL:
<http://localhost:4200>

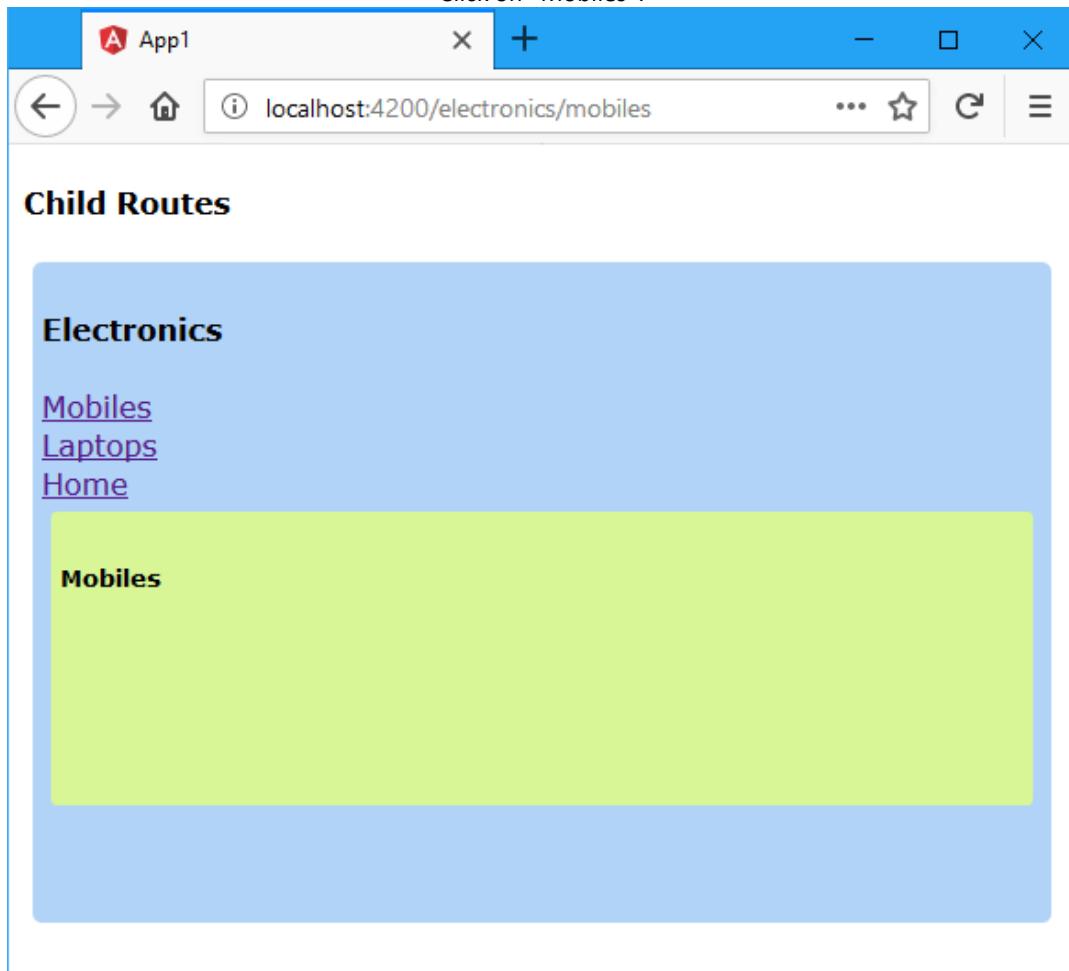


UXAII

Click on “Electronics”.



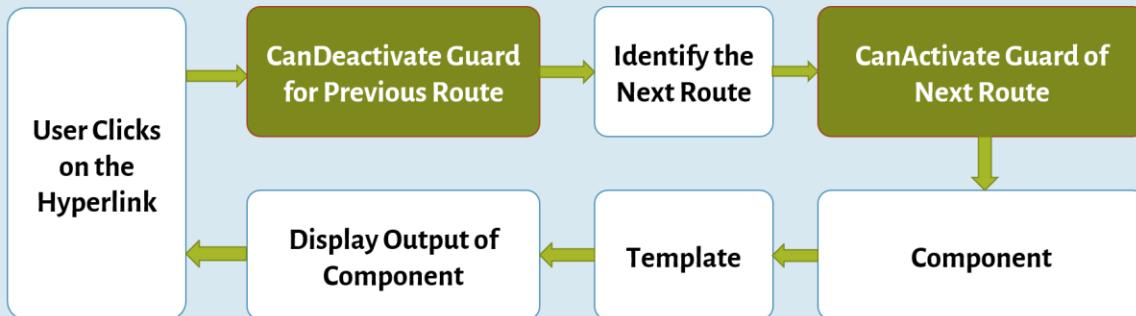
Click on “Mobiles”.



Guards

- The Guard is a service that executes at the specified situation while angular is navigating from one route to another route.
- Angular mainly supports two types of Guards:
 - **CanActivate:** Executes before entering into a route.
 - **CanDeactivate:** Executes before leaving a route.

CanActivate Guard & CanDeactivate Guard



CanActivate

- The "CanActivate" Guard executes before entering into a route.
- **Process:** User clicks on the hyperlink → Identify the route → CanActivate Guard → Navigate to the Route → Corresponding component.
- This guard can be created by implementing "CanActivate" interface.
- The "CanActivate" interface has a method called "canActivate". This method must return a boolean value, which indicates whether the route can be navigated or not. If we return "true", the route will be navigated; if we return "false", the route navigation will be stopped.
- It can receive an argument of "ActivatedRouteSnapshot" type, which represents the current state of the route.

Steps for Working with CanActivate

- Import "CanActivate" interface from "@angular/router" package:

```
import { CanActivate, ActivatedRouteSnapshot } from "@angular/router";
```

- Create a Service that implements "CanActivate" interface:

```
class Serviceclassname implements CanActivate
{
  canActivate(route: ActivatedRouteSnapshot): boolean
  {
    return true / false;
  }
}
```

- **Add service to the module:**

```
@NgModule( { ..., providers: [ Serviceclassname ] } )  
class moduleclassname  
{  
}  
}
```

- **Add guard to the route:**

```
{ path: "path here", component: ComponentClassname, canActivate: [ Serviceclassname ] }
```

CanActivate - Example

Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular  
ng new app1  
cd c:\angular\app1  
ng g component OnlineShopping  
ng g component Appliances  
ng g component Electronics  
ng g component Fashion  
ng g component Furniture  
ng g component Lighting  
ng g component Mobiles  
ng g component Laptops  
ng g component Men  
ng g component Women  
ng g component Furniture  
ng g component Login  
ng g service LoginStatus  
ng g service LoginAuth
```

c:\angular\app1\package.json

```
{  
  "name": "app1",  
  "version": "0.0.0",  
  "license": "MIT",
```

```
"scripts": {
  "ng": "ng",
  "start": "ng serve",
  "build": "ng build --prod",
  "test": "ng test",
  "lint": "ng lint",
  "e2e": "ng e2e"
},
"private": true,
"dependencies": {
  "@angular/animations": "^5.2.0",
  "@angular/common": "^5.2.0",
  "@angular/compiler": "^5.2.0",
  "@angular/core": "^5.2.0",
  "@angular/forms": "^5.2.0",
  "@angular/http": "^5.2.0",
  "@angular/platform-browser": "^5.2.0",
  "@angular/platform-browser-dynamic": "^5.2.0",
  "@angular/router": "^5.2.0",
  "core-js": "^2.4.1",
  "rxjs": "^5.5.6",
  "zone.js": "^0.8.19"
},
"devDependencies": {
  "@angular/cli": "~1.7.4",
  "@angular/compiler-cli": "^5.2.0",
  "@angular/language-service": "^5.2.0",
  "@types/jasmine": "~2.8.3",
  "@types/jasminewd2": "~2.0.2",
  "@types/node": "~6.0.60",
  "codelyzer": "^4.0.1",
  "jasmine-core": "~2.8.0",
  "jasmine-spec-reporter": "~4.2.1",
  "karma": "~2.0.0",
  "karma-chrome-launcher": "~2.2.0",
  "karma-coverage-istanbul-reporter": "^1.2.1",
  "karma-jasmine": "~1.1.0",
  "karma-jasmine-html-reporter": "^0.2.2",
  "protractor": "~5.1.2",
  "ts-node": "~4.1.0",
  "tslint": "~5.9.1",
  "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\styles.css

```
#container
{
  background-color: #b2d3f8;
  margin: 5px;
  padding: 5px;
  border-radius: 5px;
```

```
    height: 350px;
}

.container2
{
  background-color: #d8f696;
  margin: 5px;
  padding: 5px;
  border-radius: 3px;
  height: 150px;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';
import { Routes, RouterModule } from '@angular/router';
import { AppliancesComponent } from './appliances/appliances.component';
import { ElectronicsComponent } from './electronics/electronics.component';
import { FashionComponent } from './fashion/fashion.component';
import { FurnitureComponent } from './furniture/furniture.component';
import { LaptopsComponent } from './laptops/laptops.component';
import { LightingComponent } from './lighting/lighting.component';
import { MenComponent } from './men/men.component';
import { MobilesComponent } from './mobiles/mobiles.component';
import { OnlineShoppingComponent } from './online-shopping/online-shopping.component';
import { WomenComponent } from './women/women.component';
import { LoginComponent } from './login/login.component';
import { LoginStatusService } from './login-status.service';
import { LoginAuthService } from './login-auth.service';

var myroutes: Routes = [
  { path: "", component: OnlineShoppingComponent },
  {
    path: "electronics", component: ElectronicsComponent, canActivate: [LoginAuthService], children: [
      { path: "mobiles", component: MobilesComponent },
      { path: "laptops", component: LaptopsComponent },
    ]
  },
  {
    path: "appliances", component: AppliancesComponent, children: [
      { path: "lighting", component: LightingComponent },
      { path: "furniture", component: FurnitureComponent },
    ]
  },
  {
    path: "fashion", component: FashionComponent, children: [
      { path: "men", component: MenComponent },
      { path: "women", component: WomenComponent },
    ]
  }
]
```

```
};

var myroutes2 = RouterModule.forRoot(myroutes);

@NgModule({
  declarations: [
    AppComponent,
    AppliancesComponent,
    ElectronicsComponent,
    FashionComponent,
    FurnitureComponent,
    LaptopsComponent,
    LightingComponent,
    MenComponent,
    MobilesComponent,
    OnlineShoppingComponent,
    WomenComponent,
    LoginComponent
  ],
  imports: [
    BrowserModule, FormsModule, myroutes2
  ],
  providers: [ LoginStatusService, LoginAuthService ],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\login-status.service.ts

```
import { Injectable } from '@angular/core';

@Injectable()
export class LoginStatusService
{
  isLoggedIn: boolean = false;
}
```

c:\angular\app1\src\app\login-auth.service.ts

```
import { Injectable, Inject } from '@angular/core';
import { LoginStatusService } from './login-status.service';
import { Router, ActivatedRouteSnapshot } from "@angular/router";

@Injectable()
export class LoginAuthService
{
  constructor( @Inject(LoginStatusService) private loginStatusService: LoginStatusService,
  @Inject(Router) private router: Router)
  {

  }

  canActivate(route: ActivatedRouteSnapshot): boolean
  {
    if (this.loginStatusService.isLoggedIn == false)
```

```
{  
  alert("You must login to access electronics page");  
  this.router.navigateByUrl("/");  
}  
return this.loginStatusService.isLoggedIn;  
}  
}
```

c:\angular\app1\src\app\login\login.component.ts

```
import { Component, Inject } from '@angular/core';  
import { LoginStatusService } from "./login-status.service";  
  
@Component({  
  selector: 'app-login',  
  templateUrl: './login.component.html',  
  styleUrls: ['./login.component.css']  
})  
export class LoginComponent  
{  
  
  username: string = "";  
  password: string = "";  
  msg: string = "";  
  loginStatus: boolean;  
  
  constructor( @Inject(LoginStatusService) private loginStatusService: LoginStatusService)  
  {  
    this.loginStatus = this.loginStatusService.isLoggedIn;  
  }  
  
  CheckLogin(txt1)  
  {  
    if (this.username == "admin" && this.password == "manager")  
    {  
      this.msg = "Successful login";  
      this.loginStatusService.isLoggedIn = true;  
      this.loginStatus = true;  
    }  
    else  
    {  
      this.msg = "Invalid login";  
      this.loginStatusService.isLoggedIn = false;  
      this.loginStatus = false;  
      txt1.focus();  
    }  
  }  
  
  Logout()  
  {  
    this.loginStatusService.isLoggedIn = false;  
    this.loginStatus = false;  
  }  
}
```

```
}
```

c:\angular\app1\src\app\login\login.component.html

```
<div>
<div *ngIf="!loginStatus">
<form>
<h5>Login</h5>
Username: <input type="text" [(ngModel)]="username" name="username" #t1><br>
Password: <input type="password" [(ngModel)]="password" name="password"><br>
<input type="submit" value="Login" (click)="CheckLogin(t1)"><br>
{{msg}}
</form>
</div>

<div *ngIf="loginStatus">
<form>
<input type="submit" value="Logout" (click)="Logout()"><br>
</form>
</div>
</div>
```

c:\angular\app1\src\app\app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div class="class1">
<h4>canActivate</h4>
<div id="container">
  <router-outlet>
  </router-outlet>
</div>
</div>
```

c:\angular\app1\src\app\online-shopping\online-shopping.component.ts

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

@Component({
  selector: 'app-online-shopping',
  templateUrl: './online-shopping.component.html',
  styleUrls: ['./online-shopping.component.css']
})
export class OnlineShoppingComponent implements OnInit {
```

```
constructor() {}

ngOnInit()
{
}
```

c:\angular\app1\src\app\online-shopping\online-shopping.component.html

```
<div>
  <h4>Online Shopping</h4>
  <a routerLink="electronics">Electronics</a><br>
  <a routerLink="appliances">Appliances</a><br>
  <a routerLink="fashion">Fashion</a><br>
  <app-login></app-login>
</div>
```

c:\angular\app1\src\app\appliances\appliances.component.ts

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

@Component({
  selector: 'app-appliances',
  templateUrl: './appliances.component.html',
  styleUrls: ['./appliances.component.css']
})
export class AppliancesComponent implements OnInit {

  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\appliances\appliances.component.html

```
<div>
  <h4>Appliances</h4>
  <a routerLink="lighting">Lighting</a><br>
  <a routerLink="furniture">Furniture</a><br>
  <a routerLink="/">Home</a><br>
  <div class="container2">
    <router-outlet>
    </router-outlet>
  </div>
</div>
```

c:\angular\app1\src\app\electronics\electronics.component.ts

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

@Component({
  selector: 'app-electronics',
```

```
templateUrl: './electronics.component.html',
styleUrls: ['./electronics.component.css']
})
export class ElectronicsComponent implements OnInit {

constructor() {}

ngOnInit()
{
}
}
```

c:\angular\app1\src\app\electronics\electronics.component.html

```
<div>
<h4>Electronics</h4>
<a routerLink="mobiles">Mobiles</a><br>
<a routerLink="laptops">Laptops</a><br>
<a routerLink="/">Home</a><br>
<div class="container2">
<router-outlet>
</router-outlet>
</div>
</div>
```

c:\angular\app1\src\app\fashion\fashion.component.ts

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

@Component({
selector: 'app-fashion',
templateUrl: './fashion.component.html',
styleUrls: ['./fashion.component.css']
})
export class FashionComponent implements OnInit {

constructor() {}

ngOnInit()
{
}
}
```

c:\angular\app1\src\app\fashion\fashion.component.html

```
<div>
<h4>Fashion</h4>
<a routerLink="men">Men</a><br>
<a routerLink="women">Women</a><br>
<a routerLink="/">Home</a><br>
<div class="container2">
<router-outlet>
</router-outlet>
</div>
```

```
</div>
```

c:\angular\app1\src\app\furniture\furniture.component.ts

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

@Component({
  selector: 'app-furniture',
  templateUrl: './furniture.component.html',
  styleUrls: ['./furniture.component.css']
})
export class FurnitureComponent implements OnInit {

  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\furniture\furniture.component.html

```
<div>
  <h5>Furniture</h5>
</div>
```

c:\angular\app1\src\app\lighting\lighting.component.ts

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

@Component({
  selector: 'app-lighting',
  templateUrl: './lighting.component.html',
  styleUrls: ['./lighting.component.css']
})
export class LightingComponent implements OnInit {

  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\lighting\lighting.component.html

```
<div>
  <h5>Lighting</h5>
</div>
```

c:\angular\app1\src\app\mobiles\mobiles.component.ts

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

@Component({
```

```
    selector: 'app-mobiles',
    templateUrl: './mobiles.component.html',
    styleUrls: ['./mobiles.component.css']
})
export class MobilesComponent implements OnInit {

  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\mobiles\mobiles.component.html

```
<div>
  <h5>Mobiles</h5>
</div>
```

c:\angular\app1\src\app\laptops\laptops.component.ts

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

@Component({
  selector: 'app-laptops',
  templateUrl: './laptops.component.html',
  styleUrls: ['./laptops.component.css']
})
export class LaptopsComponent implements OnInit {

  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\laptops\laptops.component.html

```
<div>
  <h5>Laptops</h5>
</div>
```

c:\angular\app1\src\app\men\men.component.ts

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

@Component({
  selector: 'app-men',
  templateUrl: './men.component.html',
  styleUrls: ['./men.component.css']
})
export class MenComponent implements OnInit {

  constructor() {}
```

```
ngOnInit()
{
}
}
```

c:\angular\app1\src\app\men\men.component.html

```
<div>
<h5>Men</h5>
</div>
```

c:\angular\app1\src\app\women\women.component.ts

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

@Component({
  selector: 'app-women',
  templateUrl: './women.component.html',
  styleUrls: ['./women.component.css']
})
export class WomenComponent implements OnInit {

  constructor() {}

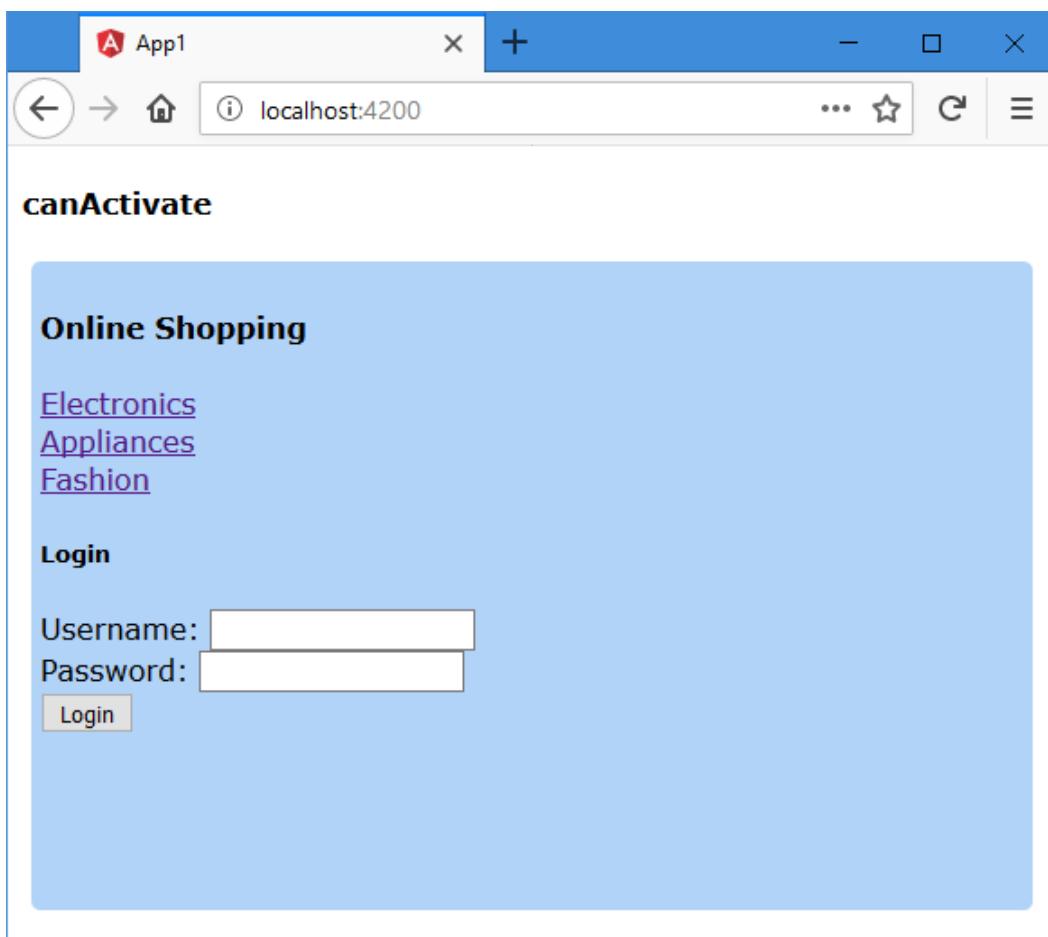
  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\women\women.component.html

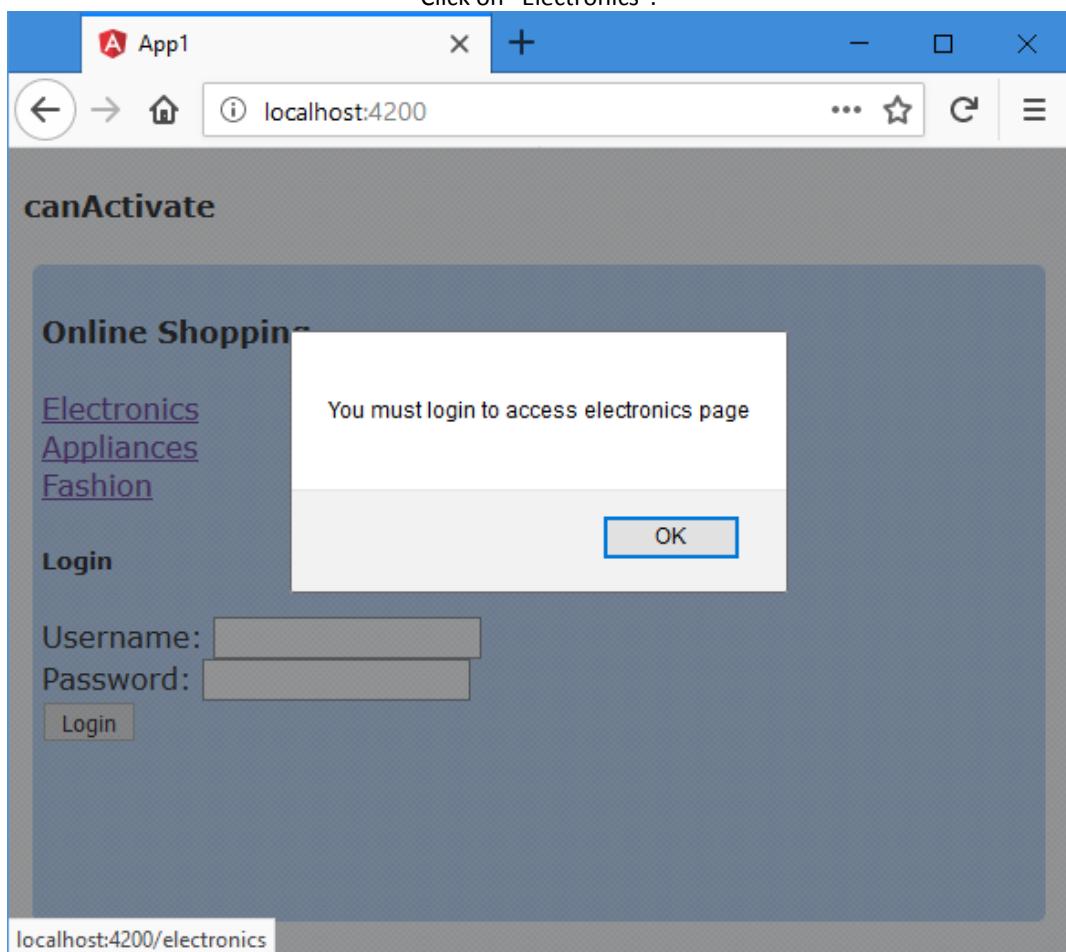
```
<div>
<h5>Women</h5>
</div>
```

Executing the application:

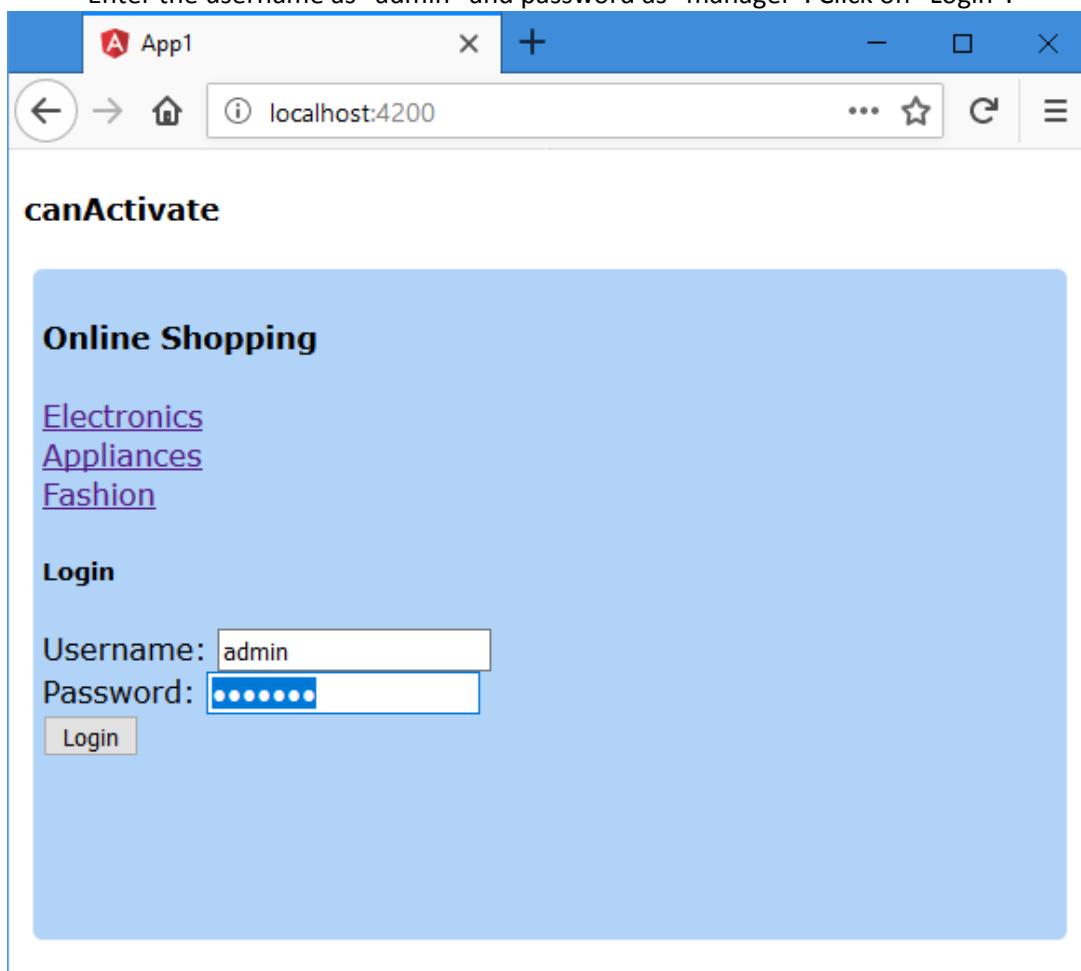
- Open Command Prompt and enter the following commands:
cd c:\angular\app1
ng serve
- Open the browser and enter the following URL:
<http://localhost:4200>

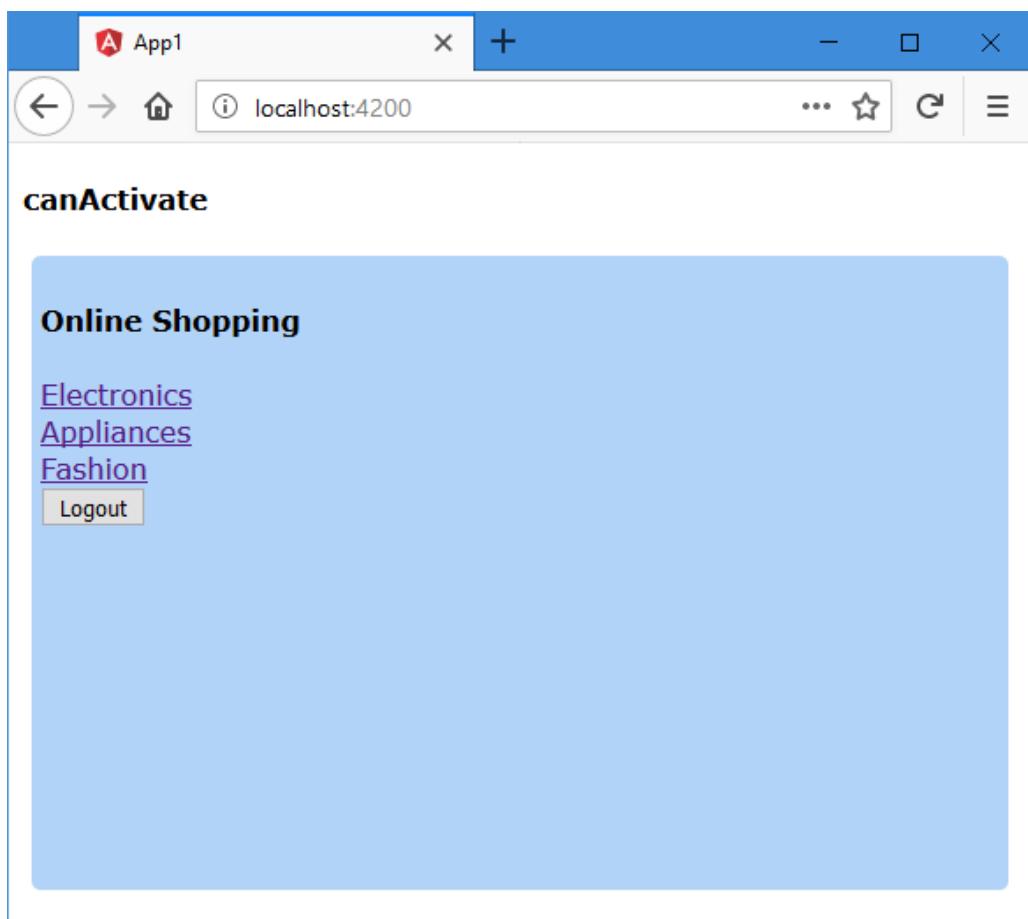


Click on “Electronics”.

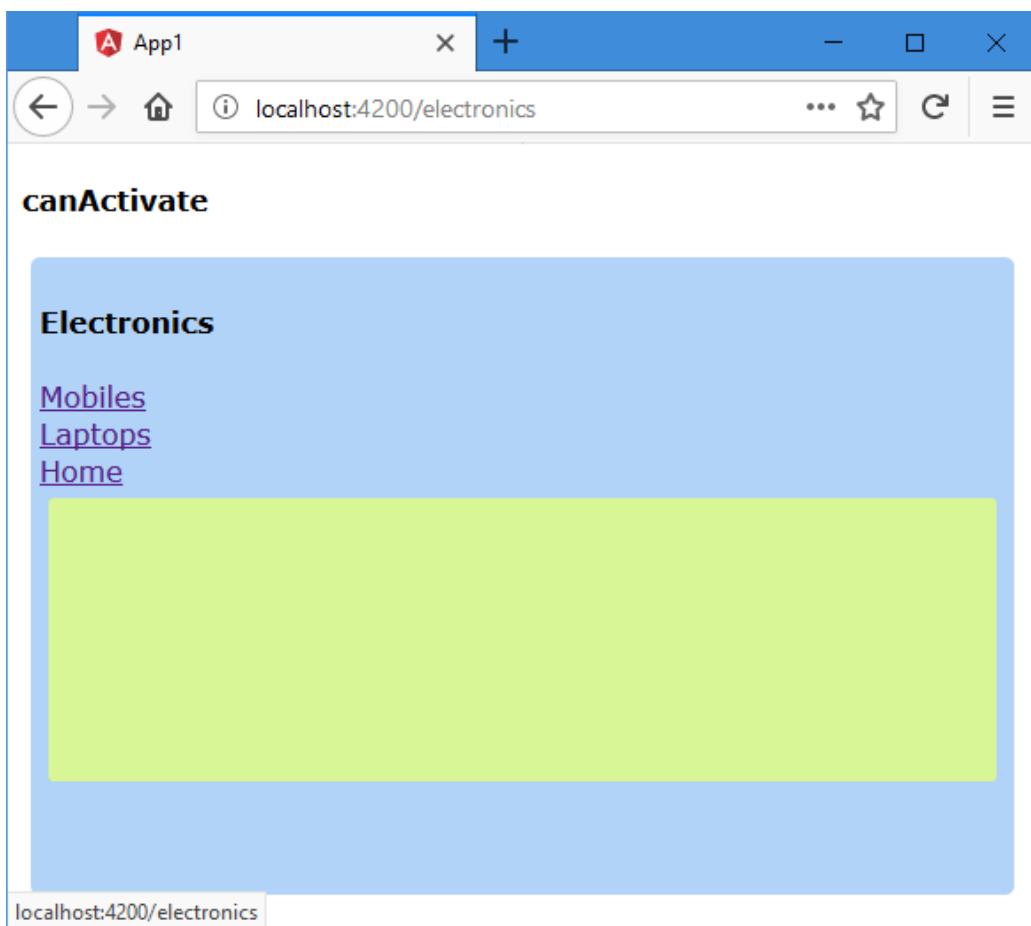


Enter the username as “admin” and password as “manager”. Click on “Login”.





Click on “Electronics” now.



CanDeactivate

- The "CanDeactivate" Guard executes before leaving from a route.
- This guard can be created by implementing "CanDeactivate" interface.
- The "CanDeactivate" interface has a method called "canDeactivate". This method must return a boolean value, which indicates whether the route can be leave or not. If we return "true", the route will be left; if we return "false", the route navigation will be stopped.
- It can receive an argument of an user-defined interface type, , which represents the current component.

Steps for Working with CanDeactive

- Import "CanDeactivate" interface from "@angular/router" package:

```
import { CanDeactivate } from "@angular/router";
```

- Create the interface for CanDeactive Guard:

```
interface interfacename  
{  
  canNavigate: boolean;  
}
```

- Create a Service that implements "CanDeactivate" interface:

```
class Serviceclassname implements CanDeactivate<interfacename>  
{  
  canDeactivate(component: interfacename): boolean  
  {  
    return true / false;  
  }  
}
```

- Add service to the module:

```
@NgModule( { ..., providers: [ Serviceclassname ] } )  
class moduleclassname  
{  
}
```

- Add guard to the route:

```
{ path: "path here", component: ComponentClassname, canDeactivate: [ Serviceclassname ] }
```

CanDeactivate - Example

Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular  
ng new app1  
cd c:\angular\app1  
ng g component Home  
ng g component About  
ng g component Contact  
ng g class CanComponentDeactivate  
ng g service CanDeactiveGuard
```

c:\angular\app1\package.json

```
{  
  "name": "app1",  
  "version": "0.0.0",  
  "license": "MIT",  
  "scripts": {  
    "ng": "ng",  
    "start": "ng serve",  
    "build": "ng build --prod",  
    "test": "ng test",  
    "lint": "ng lint",  
    "e2e": "ng e2e"  
  },  
  "private": true,  
  "dependencies": {  
    "@angular/animations": "^5.2.0",  
    "@angular/common": "^5.2.0",  
    "@angular/compiler": "^5.2.0",  
    "@angular/core": "^5.2.0",  
    "@angular/forms": "^5.2.0",  
    "@angular/http": "^5.2.0",  
    "@angular/platform-browser": "^5.2.0",  
    "@angular/platform-browser-dynamic": "^5.2.0",  
    "@angular/router": "^5.2.0",  
    "core-js": "^2.4.1",  
    "rxjs": "^5.5.6",  
    "zone.js": "^0.8.19"  
  },  
}
```

```
"devDependencies": {
  "@angular/cli": "~1.7.4",
  "@angular/compiler-cli": "^5.2.0",
  "@angular/language-service": "^5.2.0",
  "@types/jasmine": "~2.8.3",
  "@types/jasminewd2": "~2.0.2",
  "@types/node": "~6.0.60",
  "codelyzer": "^4.0.1",
  "jasmine-core": "~2.8.0",
  "jasmine-spec-reporter": "~4.2.1",
  "karma": "~2.0.0",
  "karma-chrome-launcher": "~2.2.0",
  "karma-coverage-istanbul-reporter": "^1.2.1",
  "karma-jasmine": "~1.1.0",
  "karma-jasmine-html-reporter": "^0.2.2",
  "protractor": "~5.1.2",
  "ts-node": "~4.1.0",
  "tslint": "~5.9.1",
  "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\styles.css

```
#container
{
  background-color: #ccccff;
  margin: 5px;
  padding: 5px;
  border-radius: 5px;
  height: 200px;
}
```

c:\angular\app1\src\app\can-component-deactivate.ts

```
export interface CanComponentDeactivate
{
  canNavigate: boolean;
}
```

c:\angular\app1\src\app\can-deactivate-guard.service.ts

```
import { Injectable } from '@angular/core';
import { CanDeactivate } from '@angular/router';
import { CanComponentDeactivate } from './can-component-deactivate';

@Injectable()
export class CanDeactivateGuardService implements CanDeactivate<CanComponentDeactivate>
{
  canDeactivate(component: CanComponentDeactivate)
  {
    if (component.canNavigate == true)
    {
      return true;
    }
  }
}
```

```
    }
    else
    {
        if (confirm("Do you want to discard changes?"))
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";
import { Routes, RouterModule } from "@angular/router";
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { ContactComponent } from './contact/contact.component';
import { CanDeactivateGuardService } from "./can-deactivate-guard.service";

var myroutes: Routes = [
    { path: "", component: HomeComponent },
    { path: "home", component: HomeComponent, canDeactivate: [CanDeactivateGuardService] },
    { path: "about", component: AboutComponent },
    { path: "contact", component: ContactComponent }
];
var myroutes2 = RouterModule.forRoot(myroutes);

@NgModule({
    declarations: [
        AppComponent,
        HomeComponent,
        AboutComponent,
        ContactComponent
    ],
    imports: [
        BrowserModule, FormsModule, myroutes2
    ],
    providers: [CanDeactivateGuardService ],
    bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

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

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
```

c:\angular\app1\src\app\app.component.html

```
<div class="class1">
  <h4>CanDeactivate</h4>
  <a routerLink="home">Home</a>
  <a routerLink="about">About</a>
  <a routerLink="contact">Contact</a>

  <div id="container">
    <router-outlet>
    </router-outlet>
  </div>
</div>
```

c:\angular\app1\src\app\home\home.component.ts

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

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent
{
  firstname: string = null;
  lastname: string = null;
  canNavigate: boolean = true;

  onFirstNameChange()
  {
    this.canNavigate = false;
  }

  onLastNameChange()
  {
    this.canNavigate = false;
  }

  onSave()
  {
    this.canNavigate = true;
    alert("Saved");
  }
}
```

```
}
```

c:\angular\app1\src\app\home\home.component.html

```
<div class="class1">
<h5>Home</h5>
Firstname: <input type="text" (change)="onFirstNameChange()"><br>
Lastname: <input type="text" (change)="onLastNameChange()"><br>
<input type="submit" value="Save" (click)="onSave()">
</div>
```

c:\angular\app1\src\app\about\about.component.ts

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

@Component({
  selector: 'app-about',
  templateUrl: './about.component.html',
  styleUrls: ['./about.component.css']
})
export class AboutComponent implements OnInit {

  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\about\about.component.html

```
<div class="class1">
<h5>About</h5>
</div>
```

c:\angular\app1\src\app\contact\contact.component.ts

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

@Component({
  selector: 'app-contact',
  templateUrl: './contact.component.html',
  styleUrls: ['./contact.component.css']
})
export class ContactComponent implements OnInit
{
  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\contact\contact.component.html

```
<div class="class1">
<h5>Contact</h5>
</div>
```

Executing the application:

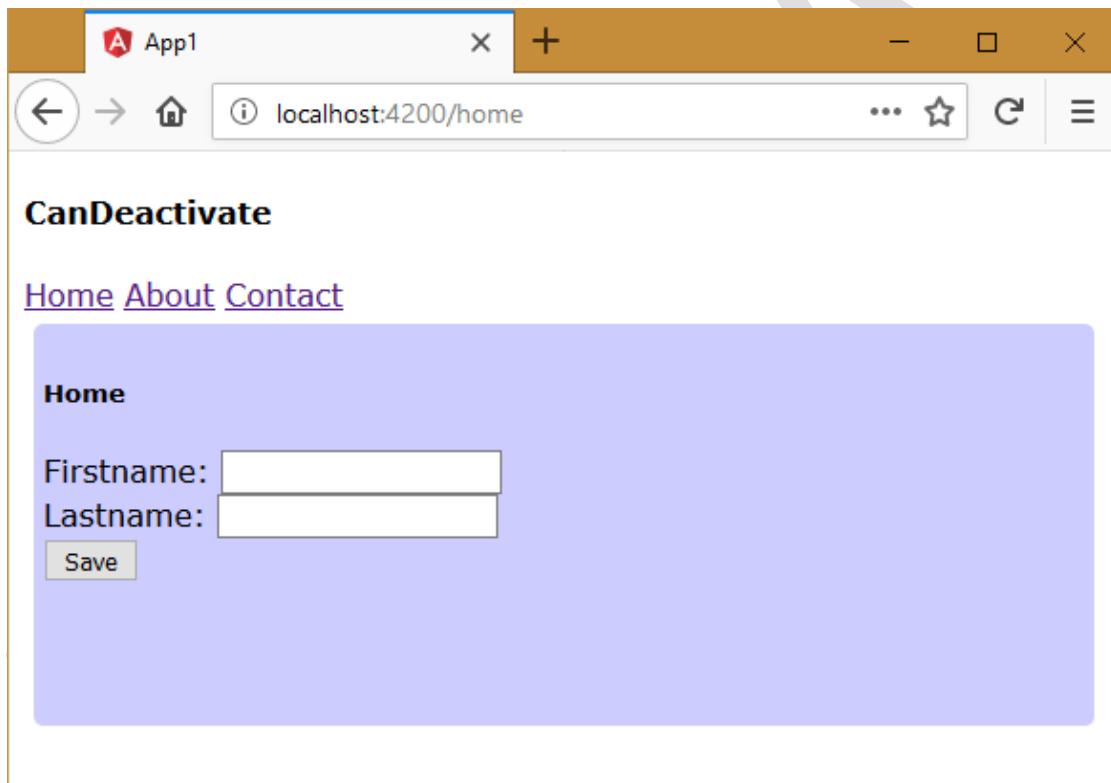
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

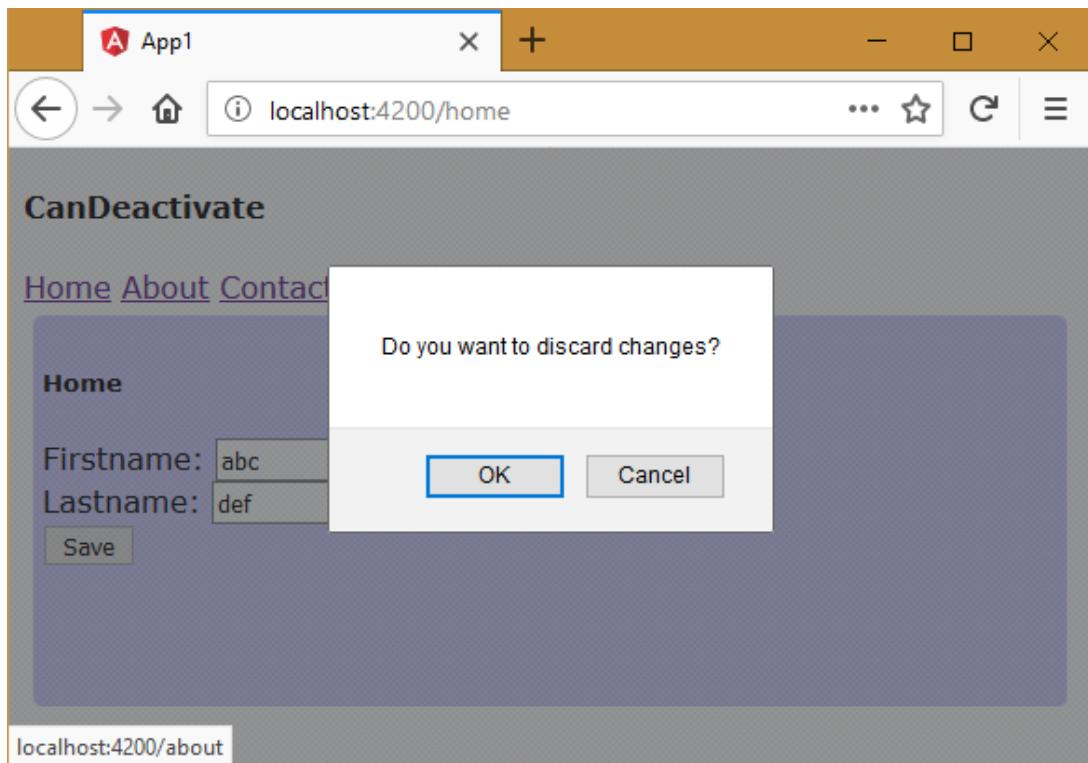
```
ng serve
```

- Open the browser and enter the following URL:

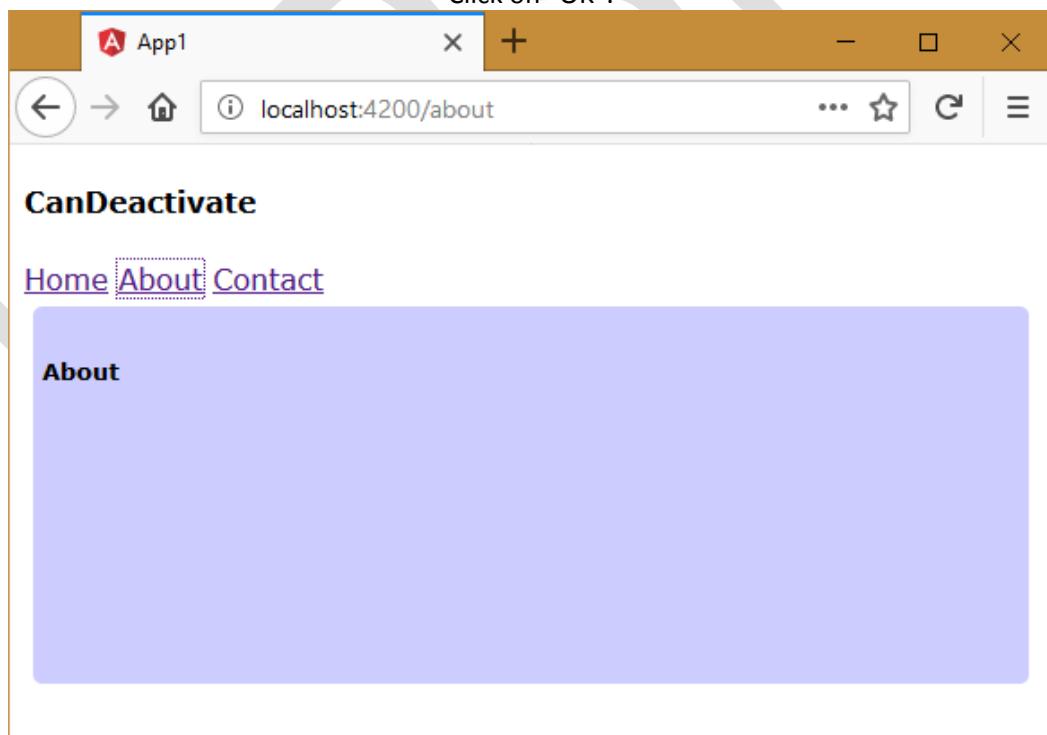
```
http://localhost:4200
```



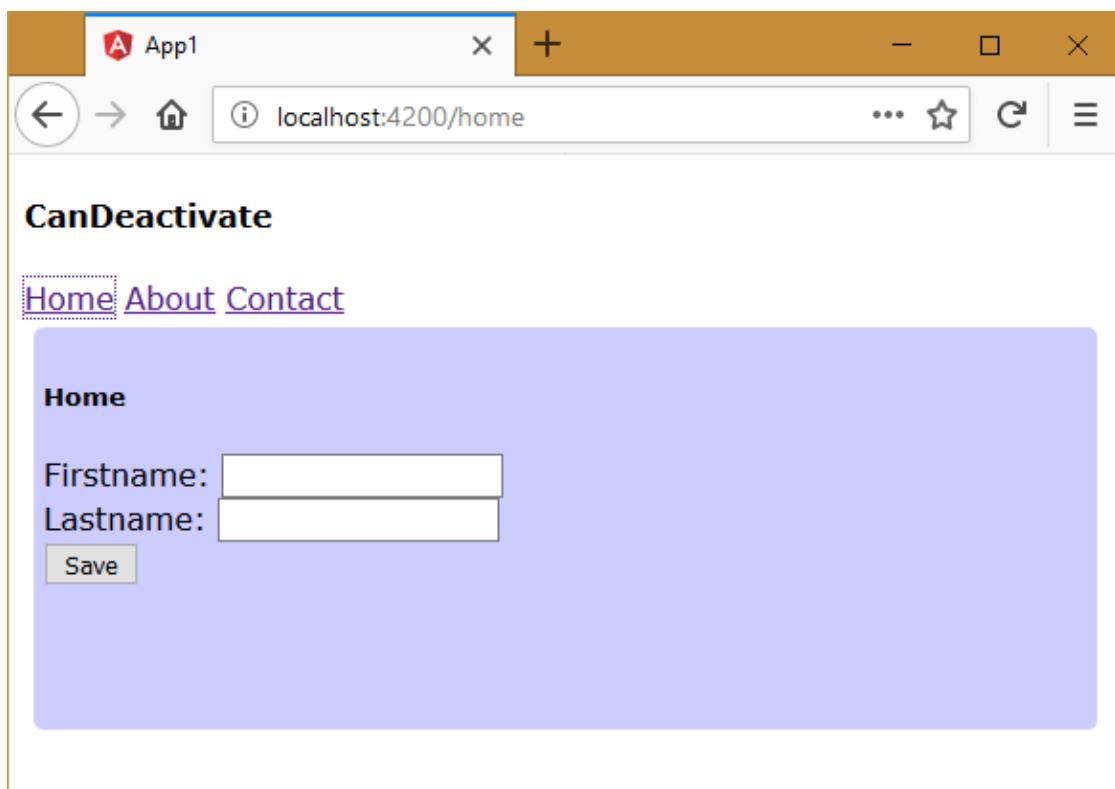
Type some firstname and lastname and click on “About”.



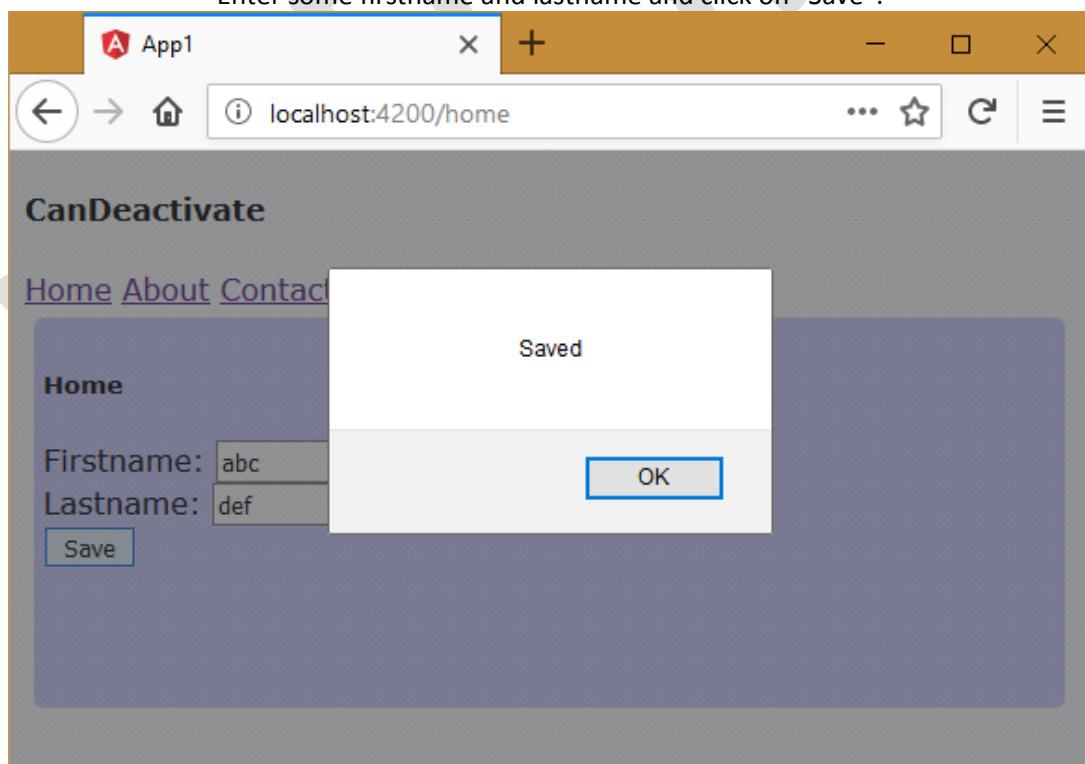
Click on "OK".



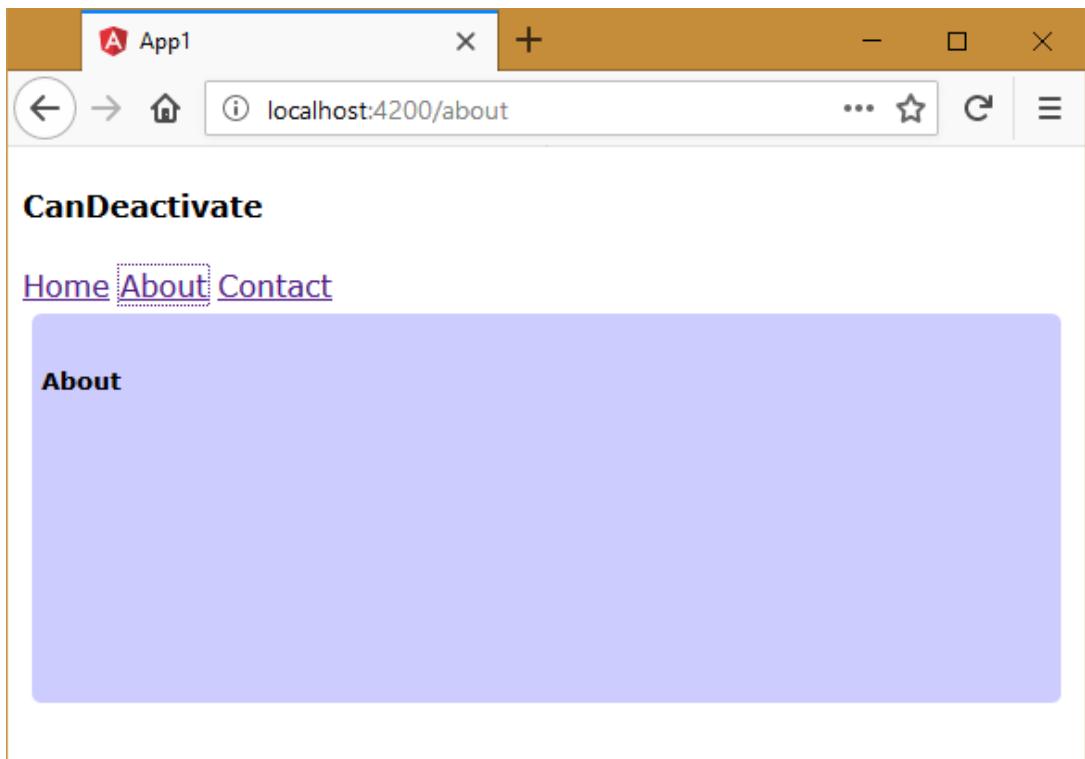
Click on "Home".



Enter some firstname and lastname and click on "Save".



Click on "About".



Deployment

Deployment to Java

Setting-up Environment for Java

- Install Java from "<https://java.com/en/download>".
- Add "C:\Program Files\Java\jdk1.8.0_172\bin" as "Path" of system variables.
- Add "JAVA_HOME" with "C:\Program Files\Java\jdk1.8.0_172" in system variables.
- Download tomcat from "<https://tomcat.apache.org/download-90.cgi>". Click on "zip" in "Core". You will get a file called "apache-tomcat-9.0.7.zip". Right click on "apache-tomcat-9.0.7.zip" and click on "Extract All". Copy all contents of the extracted folder into "c:\tomcat" folder.
- Open Command Prompt and enter the following commands:

```
cd c:\tomcat\bin  
startup.bat
```

Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular  
ng new app1
```

c:\angular\app1\package.json

{

```
"name": "app1",
"version": "0.0.0",
"license": "MIT",
"scripts": {
  "ng": "ng",
  "start": "ng serve",
  "build": "ng build --prod",
  "test": "ng test",
  "lint": "ng lint",
  "e2e": "ng e2e"
},
"private": true,
"dependencies": {
  "@angular/animations": "^5.2.0",
  "@angular/common": "^5.2.0",
  "@angular/compiler": "^5.2.0",
  "@angular/core": "^5.2.0",
  "@angular/forms": "^5.2.0",
  "@angular/http": "^5.2.0",
  "@angular/platform-browser": "^5.2.0",
  "@angular/platform-browser-dynamic": "^5.2.0",
  "@angular/router": "^5.2.0",
  "core-js": "^2.4.1",
  "rxjs": "^5.5.6",
  "zone.js": "^0.8.19"
},
"devDependencies": {
  "@angular/cli": "~1.7.4",
  "@angular/compiler-cli": "^5.2.0",
  "@angular/language-service": "^5.2.0",
  "@types/jasmine": "~2.8.3",
  "@types/jasminewd2": "~2.0.2",
  "@types/node": "~6.0.60",
  "codelyzer": "^4.0.1",
  "jasmine-core": "~2.8.0",
  "jasmine-spec-reporter": "~4.2.1",
  "karma": "~2.0.0",
  "karma-chrome-launcher": "~2.2.0",
  "karma-coverage-istanbul-reporter": "^1.2.1",
  "karma-jasmine": "~1.1.0",
  "karma-jasmine-html-reporter": "^0.2.2",
  "protractor": "~5.1.2",
  "ts-node": "~4.1.0",
  "tslint": "~5.9.1",
  "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\app.module.ts

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

```

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

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

c:\angular\app1\src\app\app.component.ts

```

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{
}

```

c:\angular\app1\src\app\app.component.html

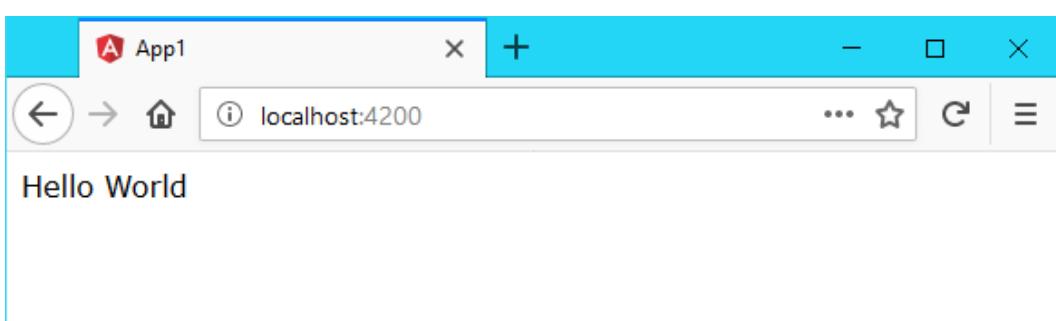
```

<div>
  Hello World
</div>

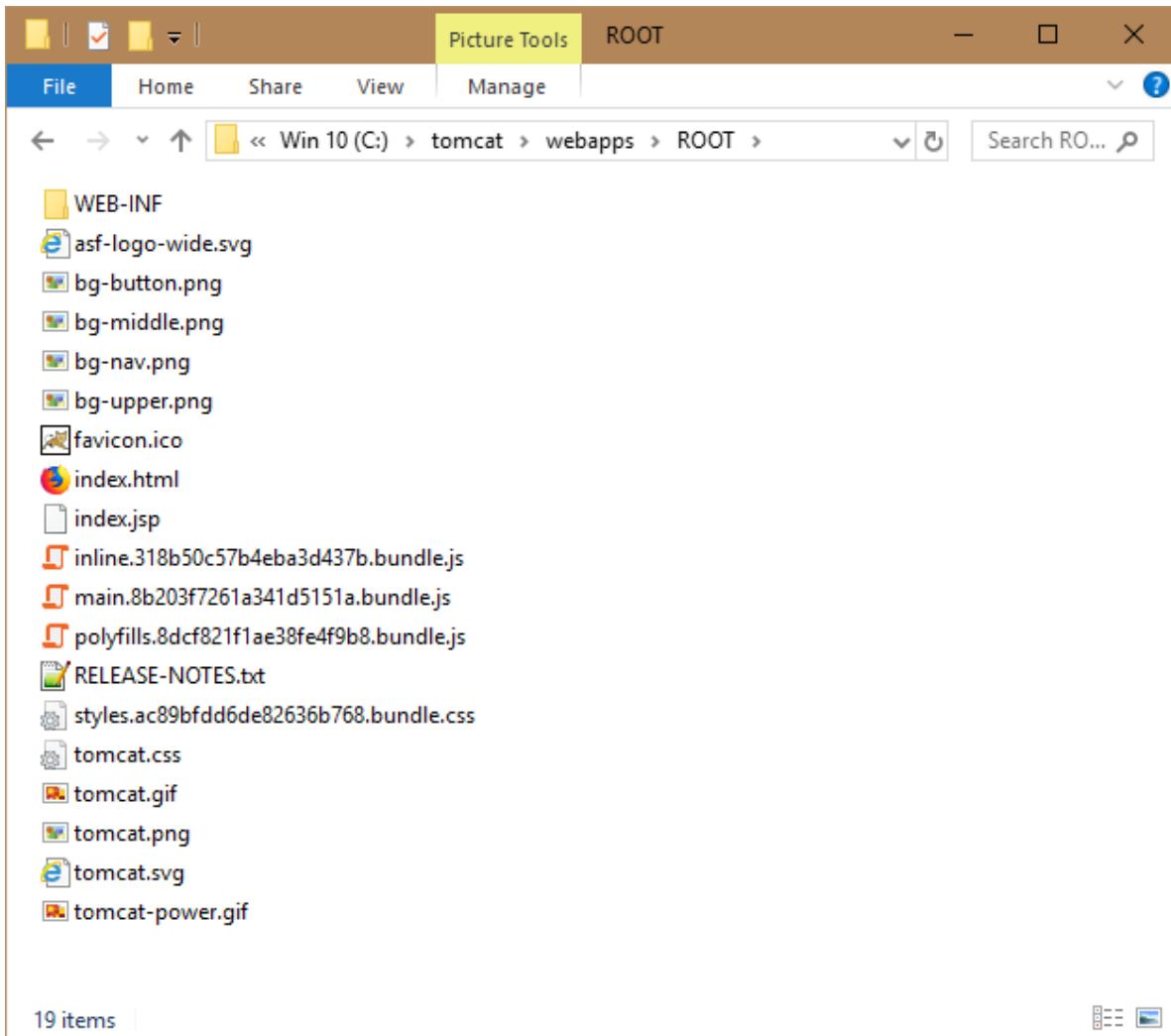
```

Executing the application:

- Open Command Prompt and enter the following commands:
 cd c:\angular\app1
 ng build --prod
 ng serve
- Open the browser and enter the following URL:
 http://localhost:4200

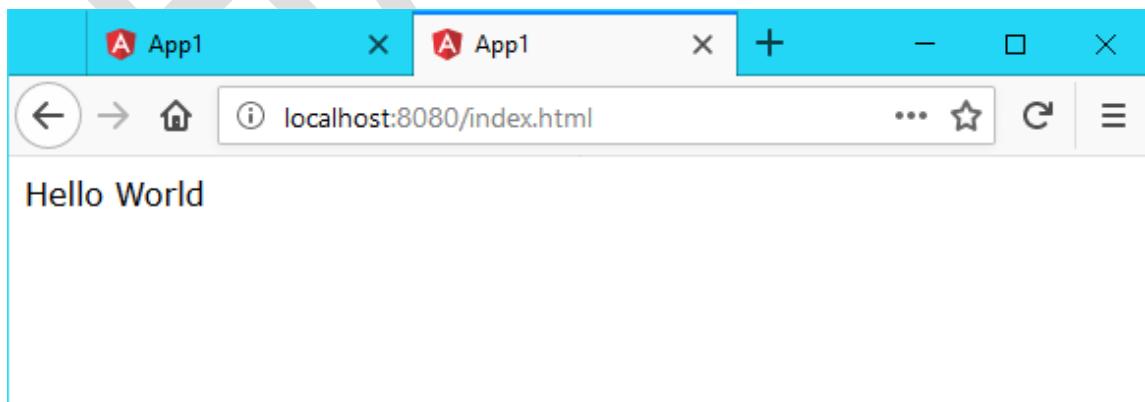


- Copy all files from “c:\angular\app1\dist” folder to “c:\tomcat\webapps\ROOT”.



- Open the browser and enter the following URL:

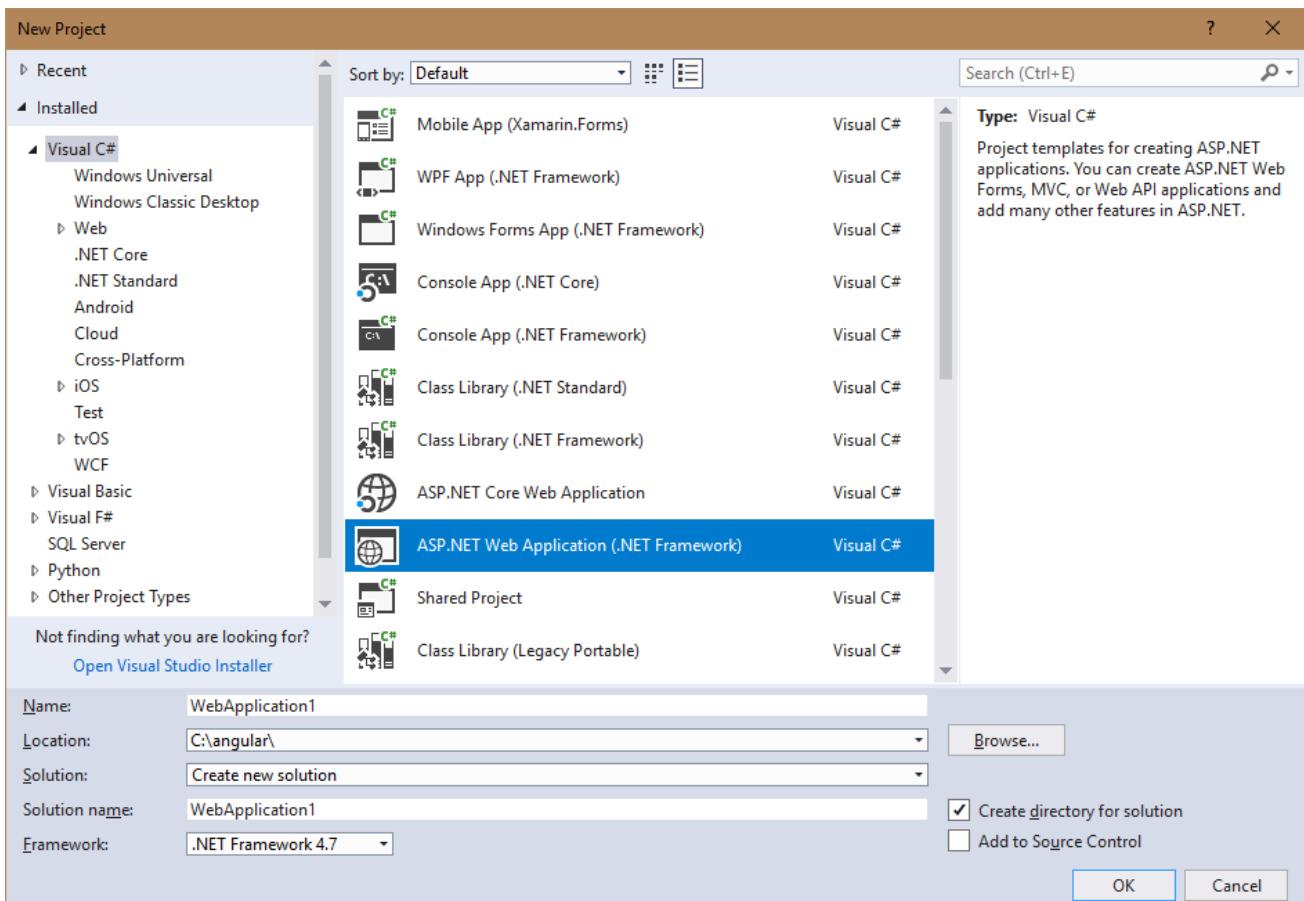
<http://localhost:8080/index.html>



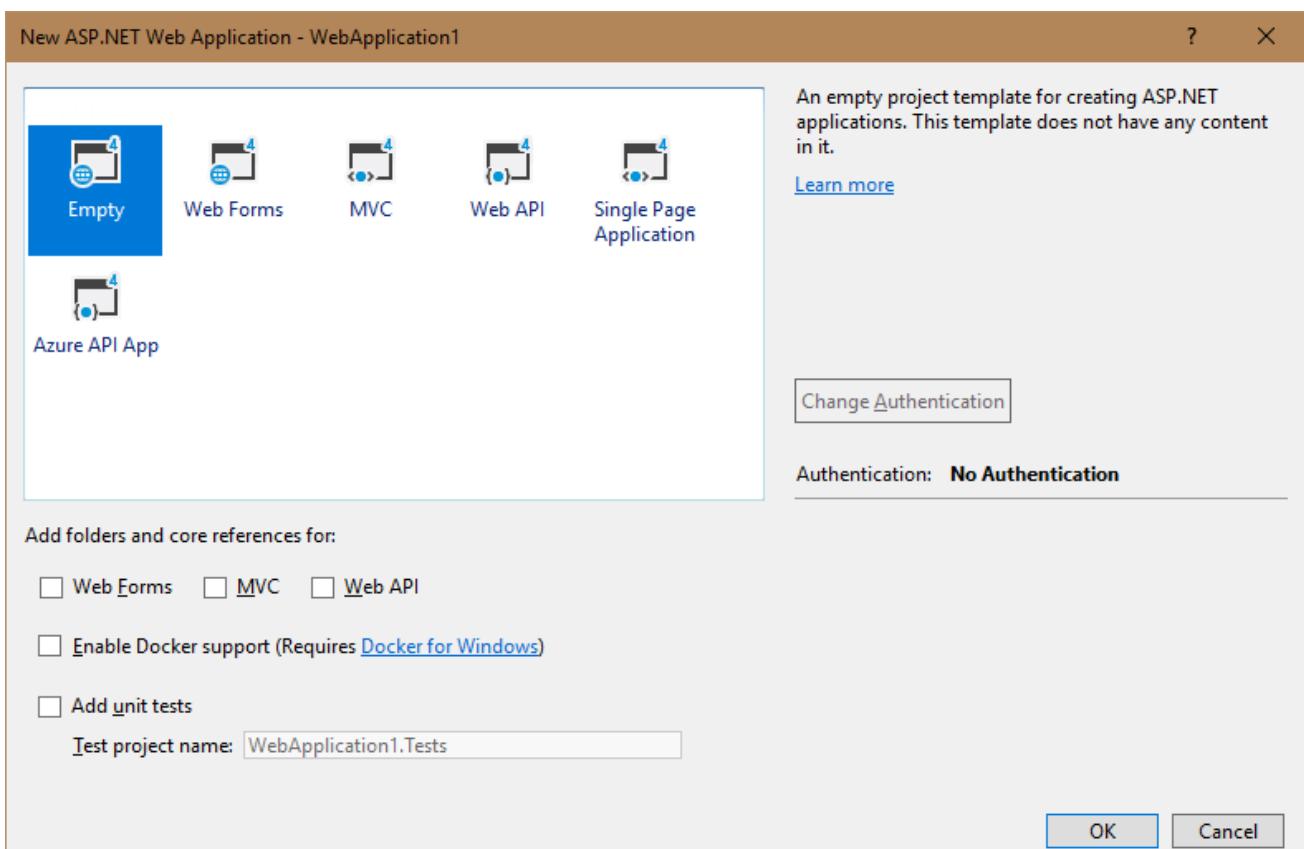
Deployment to .NET

Setting-up Environment for .NET

- Install Visual Studio Community 2017 from “<https://www.visualstudio.com>”.
- Open Visual Studio 2017. Click on “File” – “New” – “Project” – “Visual C#” – “ASP.NET Web Application (.NET Framework)”. Enter the project name “WebApplication1”. Enter the location as “c:\angular”. Click on “OK”.



- Click on “Empty”. Click on OK.



Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
```

```
ng new app1
```

c:\angular\app1\package.json

```
{
  "name": "app1",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build --prod",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^5.2.0",
    "@angular/common": "^5.2.0",
    "@angular/compiler": "^5.2.0",
    "@angular/core": "^5.2.0",
    "@angular/forms": "^5.2.0",
    "@angular/http": "^5.2.0",
    "@angular/platform-browser": "^5.2.0",
    "@angular/platform-browser-dynamic": "^5.2.0",
    "@angular/router": "^5.2.0"
  }
}
```

```
"@angular/forms": "^5.2.0",
"@angular/http": "^5.2.0",
"@angular/platform-browser": "^5.2.0",
"@angular/platform-browser-dynamic": "^5.2.0",
"@angular/router": "^5.2.0",
"core-js": "^2.4.1",
"rxjs": "^5.5.6",
"zone.js": "^0.8.19"
},
"devDependencies": {
  "@angular/cli": "~1.7.4",
  "@angular/compiler-cli": "^5.2.0",
  "@angular/language-service": "^5.2.0",
  "@types/jasmine": "~2.8.3",
  "@types/jasminewd2": "~2.0.2",
  "@types/node": "~6.0.60",
  "codelyzer": "^4.0.1",
  "jasmine-core": "~2.8.0",
  "jasmine-spec-reporter": "~4.2.1",
  "karma": "~2.0.0",
  "karma-chrome-launcher": "~2.2.0",
  "karma-coverage-istanbul-reporter": "^1.2.1",
  "karma-jasmine": "~1.1.0",
  "karma-jasmine-html-reporter": "^0.2.2",
  "protractor": "~5.1.2",
  "ts-node": "~4.1.0",
  "tslint": "~5.9.1",
  "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\app.module.ts

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

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

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

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

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div>
  Hello World
</div>
```

Executing the application:

- Open Command Prompt and enter the following commands:

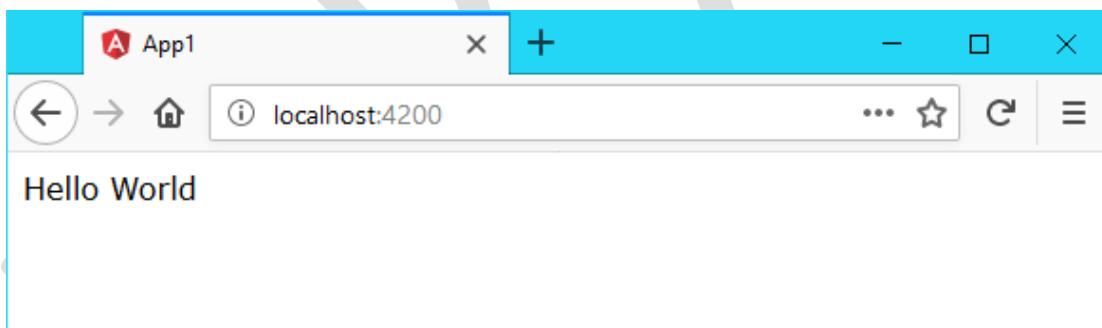
```
cd c:\angular\app1
```

```
ng build --prod
```

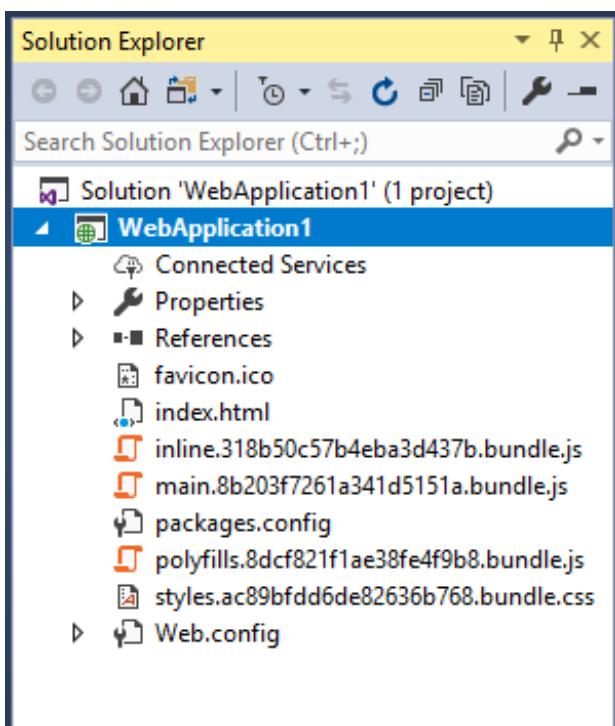
```
ng serve
```

- Open the browser and enter the following URL:

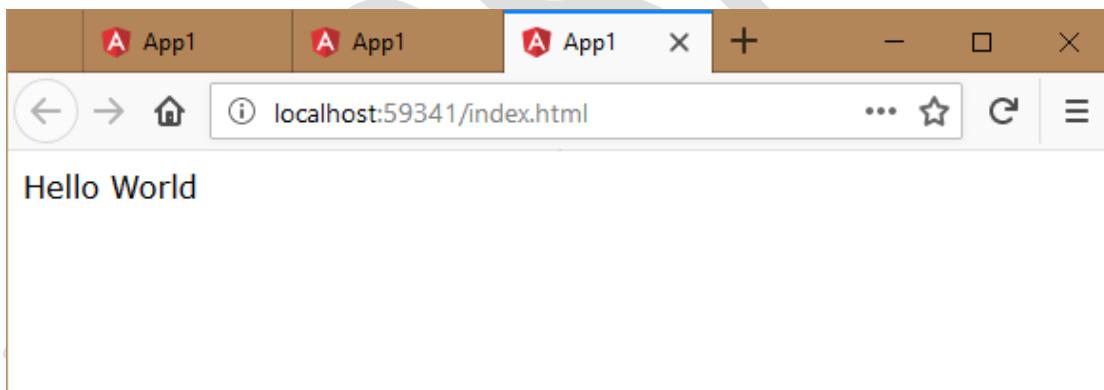
```
http://localhost:4200
```



- Copy all files from "c:\angular\app1\dist" folder to "Solution Explorer".



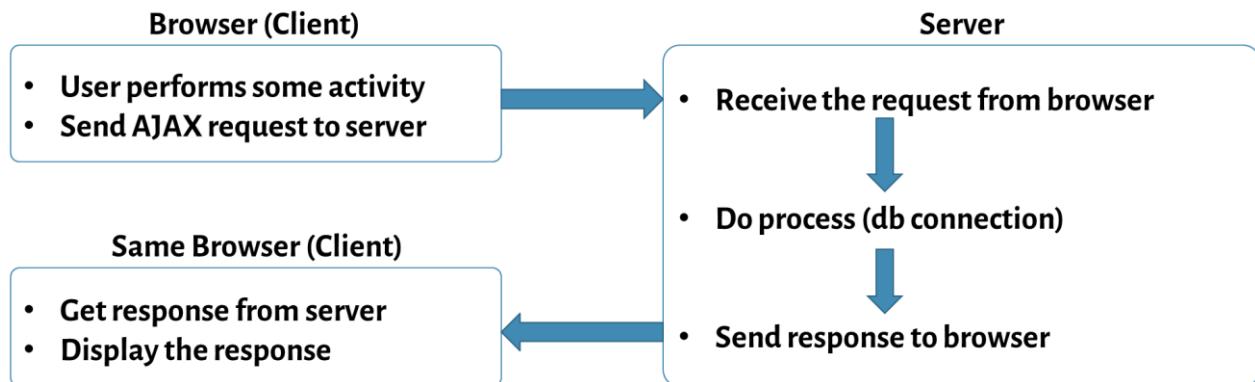
- Right click on "index.html" and click on "View in Browser".



AJAX

- AJAX (Asynchronous JavaScript And Xml) is not a language, but it is a “concept”, which is used to “send a background request from browser to server” and also “get background response from server to browser”, without refreshing (reloading) the web page in the browser.
- AJAX allows us to interact with the server and get some data from server, without refreshing the full web page.
- Ex: Facebook like button, comments, IRCTC search trains.

Execution Flow of AJAX



Advantages of AJAX

- Executes faster
- Less burden on browser (client) and server
- Better user experience.

Types of AJAX Request

- Get : Used to retrieve / search data from server
- Post : Used to insert data to server.
- Put : Used to update data on server.
- Delete : Used to delete data from server

“@angular/common/http” package

- The “@angular/common/http” package provides necessary services to send ajax request to server and get ajax response from server.

Steps for working with “@angular/common/http” package:

- Import “@angular/common” package in “package.json”:

```

"dependencies": {
  "@angular/common": "latest"
}
  
```

- Import “HttpClientModule” module:

```
import { HttpClientModule, HttpClient } from "@angular/common/http";
```

- Import “HttpClientModule” in “AppModule”:

```
@NgModule( { ..., imports: [ ..., HttpClientModule ] } )
class AppModule
```

```
{  
}
```

- **Import “HttpClient” service:**

```
import { HttpClient } from "@angular/common/http";
```

- **Get “HttpClient” service in “AppComponent”:**

```
constructor(@Inject(HttpClient) private http : HttpClient)  
{  
}
```

- **Send “get” request to server:**

```
this.http.get<modelclassname>("url", { responseType: "json | text" }).subscribe(this.successcallback,  
this.errorcallback);
```

- **Send “post” request to server:**

```
this.http.post("url", { data }, { responseType: "json | text" }).subscribe(this.successcallback,  
this.errorcallback);
```

- **Send “put” request to server:**

```
this.http.put("url", { data }, { responseType: "json | text" }).subscribe(this.successcallback,  
this.errorcallback);
```

- **Send “delete” request to server:**

```
this.http.delete("url", { responseType: "json | text" }).subscribe(this.successcallback,  
this.errorcallback);
```

- **Define “success” callback function:**

```
successcallback = (response) =>  
{  
    //do something with response  
}
```

- **Define “error” callback function:**

```
errorcallback = (error) =>  
{  
    //do something with error  
}
```

AJAX – Java – Simple - Example

Setting-up Environment for Java

- Install Java from “<https://java.com/en/download>”.
- Add “C:\Program Files\Java\jdk1.8.0_172\bin” as “Path” of system variables.
- Add “JAVA_HOME” with “C:\Program Files\Java\jdk1.8.0_172” in system variables.
- Download tomcat from “<https://tomcat.apache.org/download-90.cgi>”. Click on “zip” in “Core”. You will get a file called “apache-tomcat-9.0.7.zip”. Right click on “apache-tomcat-9.0.7.zip” and click on “Extract All”. Copy all contents of the extracted folder into “c:\tomcat” folder.
- Open Command Prompt and enter the following commands:
cd c:\tomcat\bin
startup.bat

c:\tomcat\webapps\ROOT\WEB-INF\classes\SampleServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SampleServlet extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException,IOException
    {
        PrintWriter out = response.getWriter();
        out.println("Hello from server at " + new java.util.Date());
    }
}
```

c:\tomcat\webapps\ROOT\WEB-INF\web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
  version="4.0"
  metadata-complete="true">

  <display-name>Welcome to Tomcat</display-name>
  <description>
    Welcome to Tomcat
  </description>

  <servlet>
    <servlet-name>SampleServlet</servlet-name>
    <servlet-class>SampleServlet</servlet-class>
  </servlet>
```

```
<servlet-mapping>
  <servlet-name>SampleServlet</servlet-name>
  <url-pattern>/SampleServlet</url-pattern>
</servlet-mapping>

</web-app>
```

Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
```

c:\angular\app1\package.json

```
{
  "name": "app1",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build --prod",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^5.2.0",
    "@angular/common": "^5.2.0",
    "@angular/compiler": "^5.2.0",
    "@angular/core": "^5.2.0",
    "@angular/forms": "^5.2.0",
    "@angular/http": "^5.2.0",
    "@angular/platform-browser": "^5.2.0",
    "@angular/platform-browser-dynamic": "^5.2.0",
    "@angular/router": "^5.2.0",
    "core-js": "^2.4.1",
    "rxjs": "^5.5.6",
    "zone.js": "^0.8.19"
  },
  "devDependencies": {
    "@angular/cli": "~1.7.4",
    "@angular/compiler-cli": "^5.2.0",
    "@angular/language-service": "^5.2.0",
    "@types/jasmine": "~2.8.3",
    "@types/jasminewd2": "~2.0.2",
    "@types/node": "~6.0.60",
    "codelyzer": "^4.0.1",
    "jasmine-core": "~2.8.0",
    "jasmine-spec-reporter": "~4.1.0",
    "karma": "~1.7.1",
    "karma-chrome-launcher": "~2.2.0",
    "karma-jasmine": "~1.1.0",
    "karma-jasmine-html-reporter": "~0.2.2",
    "protractor": "~5.4.0",
    "ts-node": "~3.4.2",
    "tslint": "~5.9.1"
  }
}
```

```
"jasmine-spec-reporter": "~4.2.1",
"karma": "~2.0.0",
"karma-chrome-launcher": "~2.2.0",
"karma-coverage-istanbul-reporter": "^1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';

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

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule, FormsModule, HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component, Inject } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{
  message: string;

  constructor(@Inject(HttpClient) private http: HttpClient)
  {
  }

  onGetDataClick()
```

```

{
  this.http.get("/SampleServlet", { responseType: "text" }).subscribe(this.onAjaxSuccess, this.onAjaxError);
}

onAjaxSuccess = (response) =>
{
  this.message = response;
}

onAjaxError = (error) =>
{
  alert(error);
}

```

c:\angular\app1\src\app\app.component.html

```

<div>
  <h4>Ajax - Java - Simple</h4>
  <input type="button" value="Get Data from Server" (click)="onGetDataClick()">
    <div>{{message}}</div>
</div>

```

Executing the application:

- Open Command Prompt and enter the following commands:

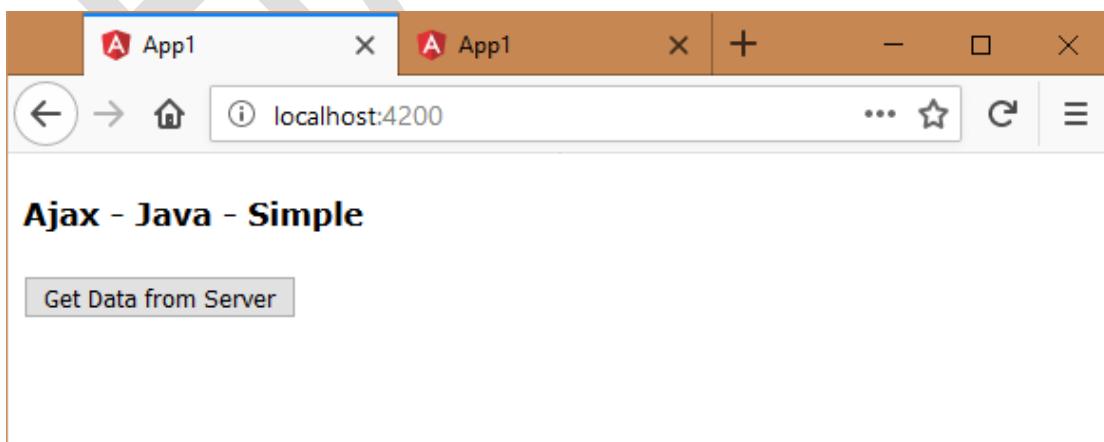
```

cd c:\tomcat\webapps\ROOT\WEB-INF\classes
javac -cp c:\tomcat\lib\servlet-api.jar SampleServlet.java
cd c:\angular\app1
ng build --prod
ng serve

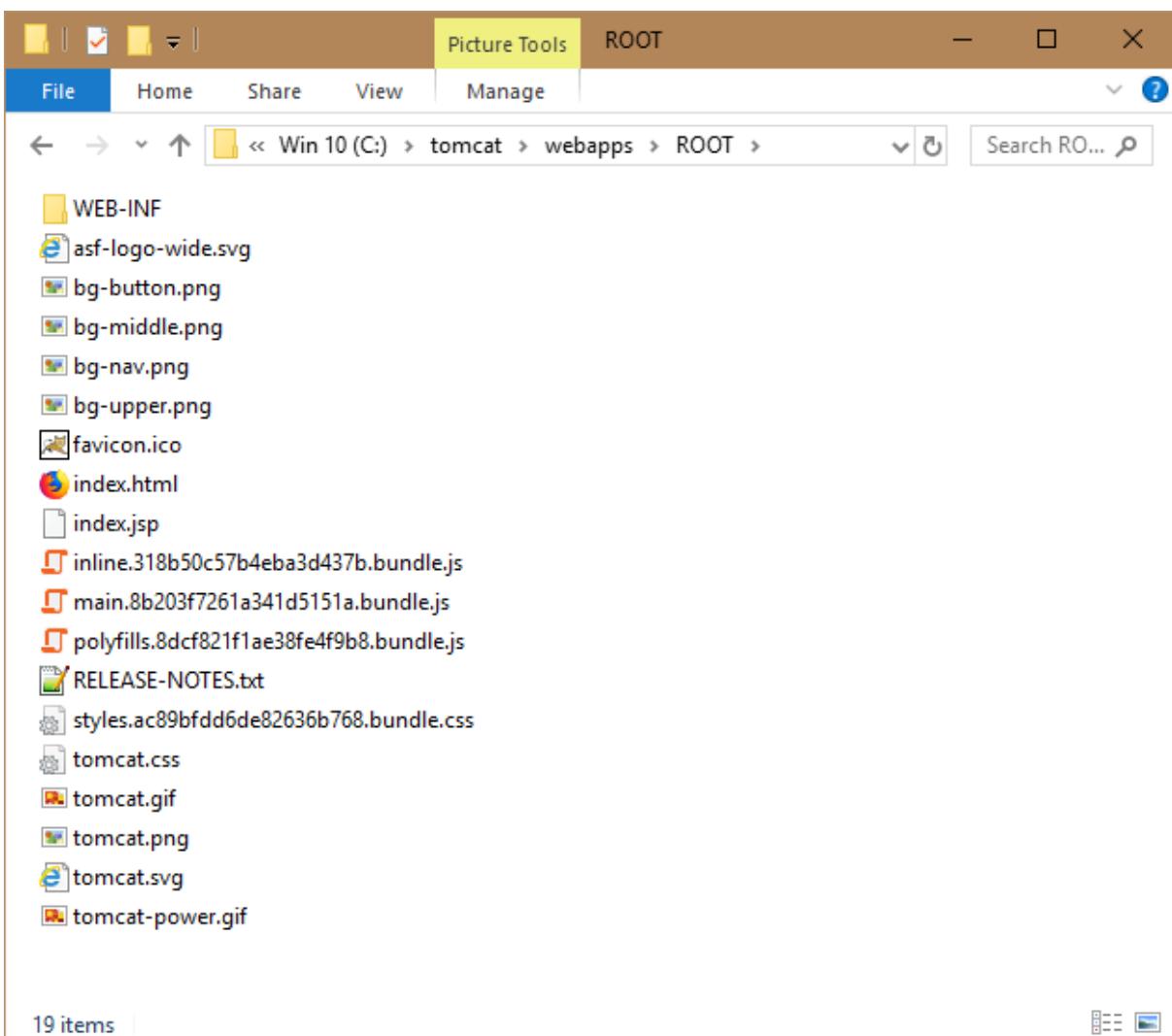
```

- Open the browser and enter the following URL:

<http://localhost:4200>

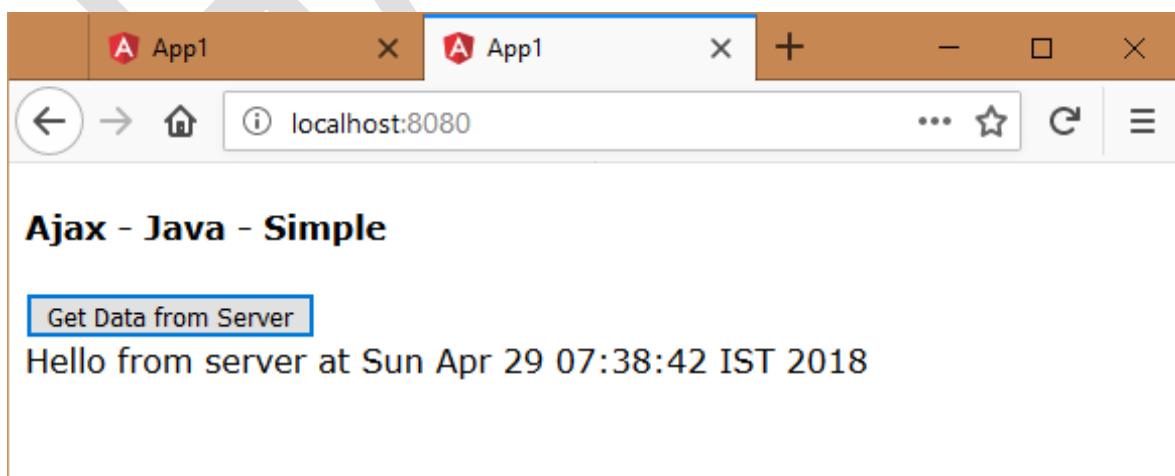


- Copy all files from “c:\angular\app1\dist” folder to “c:\tomcat\webapps\ROOT”.



- Open the browser and enter the following URL:

<http://localhost:8080/index.html>



Click on "Get Data from Server".

AJAX – Java – Get - Example

Setting-up Environment for Java

- Install Java from “<https://java.com/en/download>”.
- Add “C:\Program Files\Java\jdk1.8.0_172\bin” as “Path” of system variables.
- Add “JAVA_HOME” with “C:\Program Files\Java\jdk1.8.0_172” in system variables.
- Download tomcat from “<https://tomcat.apache.org/download-90.cgi>”. Click on “zip” in “Core”. You will get a file called “apache-tomcat-9.0.7.zip”. Right click on “apache-tomcat-9.0.7.zip” and click on “Extract All”. Copy all contents of the extracted folder into “c:\tomcat” folder.
- Open Command Prompt and enter the following commands:
cd c:\tomcat\bin
startup.bat

c:\tomcat\webapps\ROOT\WEB-INF\classes\SampleServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SampleServlet extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException,IOException
    {
        PrintWriter out = response.getWriter();
        out.println("[ { \"empid\": 1, \"empname\": \"Scott\", \"salary\": 4000 }, { \"empid\": 2, \"empname\":
\"Allen\", \"salary\": 7500 }, { \"empid\": 3, \"empname\": \"Jones\", \"salary\": 9200 }, { \"empid\": 4,
\"empname\": \"James\", \"salary\": 8400 }, { \"empid\": 5, \"empname\": \"Smith\", \"salary\": 5600 } ]");
    }
}
```

c:\tomcat\webapps\ROOT\WEB-INF\web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
  version="4.0"
  metadata-complete="true">

  <display-name>Welcome to Tomcat</display-name>
  <description>
    Welcome to Tomcat
  </description>

  <servlet>
    <servlet-name>SampleServlet</servlet-name>
    <servlet-class>SampleServlet</servlet-class>
  </servlet>
```

```
<servlet-mapping>
  < servlet-name>SampleServlet</servlet-name>
  < url-pattern>/SampleServlet</url-pattern>
</servlet-mapping>

</web-app>
```

Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g class Employee
```

c:\angular\app1\package.json

```
{
  "name": "app1",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build --prod",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^5.2.0",
    "@angular/common": "^5.2.0",
    "@angular/compiler": "^5.2.0",
    "@angular/core": "^5.2.0",
    "@angular/forms": "^5.2.0",
    "@angular/http": "^5.2.0",
    "@angular/platform-browser": "^5.2.0",
    "@angular/platform-browser-dynamic": "^5.2.0",
    "@angular/router": "^5.2.0",
    "core-js": "^2.4.1",
    "rxjs": "^5.5.6",
    "zone.js": "^0.8.19"
  },
  "devDependencies": {
    "@angular/cli": "~1.7.4",
    "@angular/compiler-cli": "^5.2.0",
    "@angular/language-service": "^5.2.0",
    "@types/jasmine": "~2.8.3",
    "@types/jasminewd2": "~2.0.2",
    "codelyzer": "3.1.2"
  }
}
```

```
"@types/node": "~6.0.60",
"codelyzer": "^4.0.1",
"jasmine-core": "~2.8.0",
"jasmine-spec-reporter": "~4.2.1",
"karma": "~2.0.0",
"karma-chrome-launcher": "~2.2.0",
"karma-coverage-istanbul-reporter": "^1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\employee.ts

```
export class Employee
{
  empid: number;
  empname: string;
  salary: number;

  constructor(a, b, c)
  {
    this.empid = a;
    this.empname = b;
    this.salary = c;
  }
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from "@angular/forms";
import { HttpClientModule } from "@angular/common/http";

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

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule, FormsModule, HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```

import { Component, Inject } from '@angular/core';
import { HttpClient } from "@angular/common/http";
import { Employee } from "./employee";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{
  employees: Employee[ ] = [];

  constructor( @Inject(HttpClient) private http: HttpClient)
  {

  }

  onGetDataClick()
  {
    this.http.get<Employee>("/SampleServlet", { responseType: "json" }).subscribe(this.onAjaxSuccess,
    this.onAjaxError);
  }

  onAjaxSuccess = (response) =>
  {
    this.employees = response;
  }

  onAjaxError = () =>
  {
    alert("error");
  }
}

```

c:\angular\app1\src\app\app.component.html

```

<div>
  <h4>Ajax - Java - Get</h4>
  <input type="button" value="Get Data" (click)="onGetDataClick()">
  <table border="1">
    <tr>
      <th>Emp ID</th>
      <th>Emp Name</th>
      <th>Salary</th>
    </tr>
    <tr *ngFor="let employee of employees">
      <td>{{employee.empid}}</td>
      <td>{{employee.empname}}</td>
      <td>{{employee.salary}}</td>
    </tr>
  </table>

```

</div>

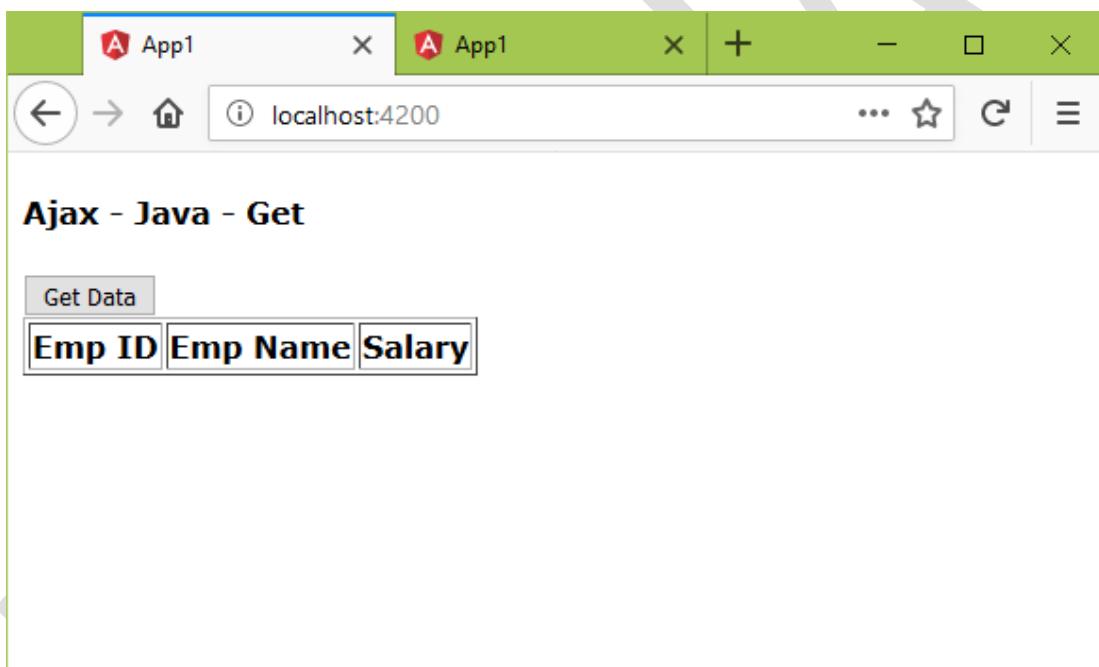
Executing the application:

- Open Command Prompt and enter the following commands:

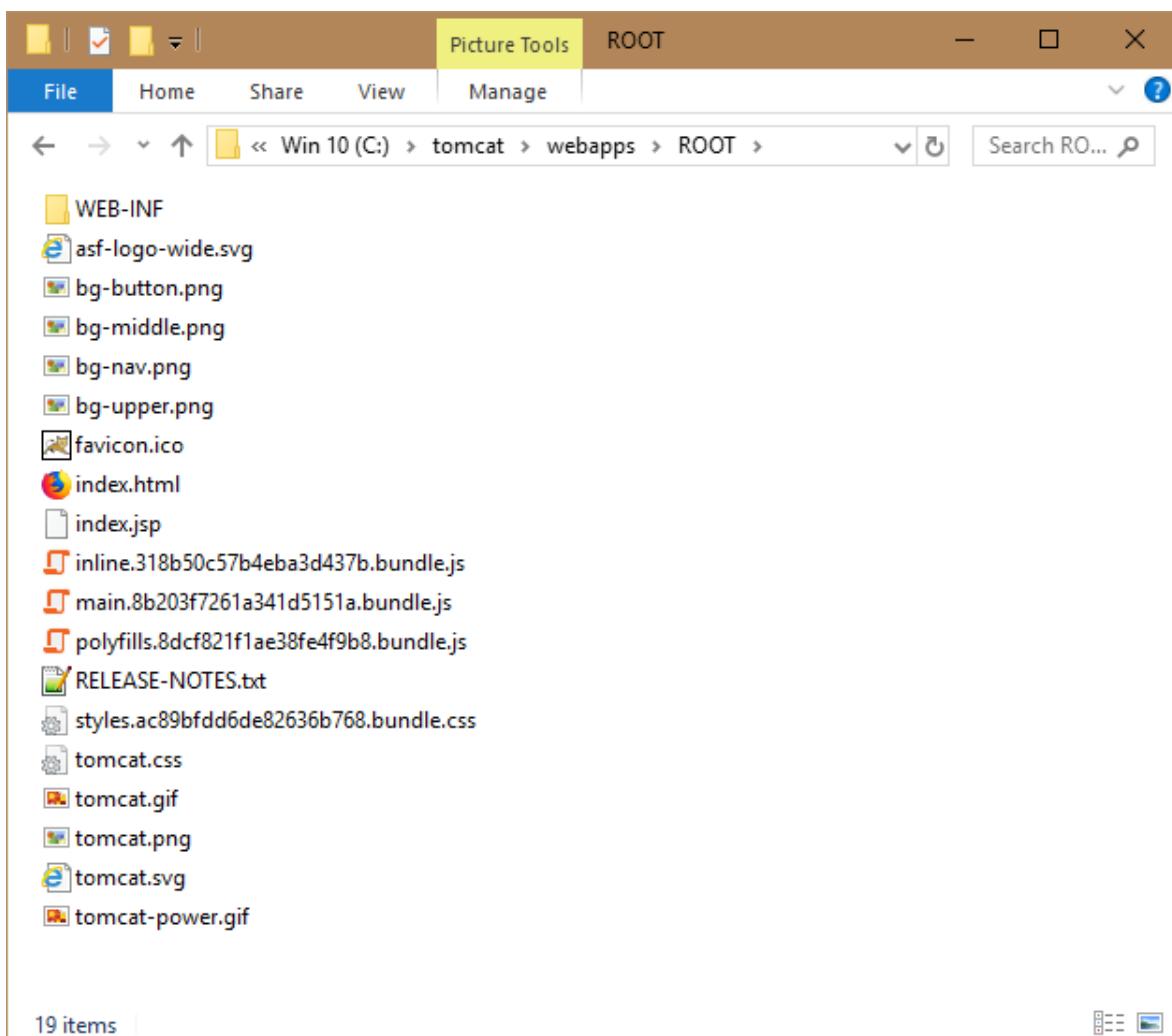
```
cd c:\tomcat\webapps\ROOT\WEB-INF\classes  
javac -cp c:\tomcat\lib\servlet-api.jar SampleServlet.java  
cd c:\angular\app1  
ng build --prod  
ng serve
```

- Open the browser and enter the following URL:

<http://localhost:4200>



- Copy all files from "c:\angular\app1\dist" folder to "c:\tomcat\webapps\ROOT".



- Open the browser and enter the following URL:

<http://localhost:8080/index.html>

The screenshot shows a web browser window with the following details:

- Title Bar:** App1
- Address Bar:** localhost:8080
- Page Content:**

Ajax - Java - Get

Get Data

| Emp ID | Emp Name | Salary |
|--------|----------|--------|
| 1 | Scott | 4000 |
| 2 | Allen | 7500 |
| 3 | Jones | 9200 |
| 4 | James | 8400 |
| 5 | Smith | 5600 |

Click on “Get Data”.

AJAX – Java – Search - Example

Setting-up Environment for Java

- Install Java from “<https://java.com/en/download>”.
- Add “C:\Program Files\Java\jdk1.8.0_172\bin” as “Path” of system variables.
- Add “JAVA_HOME” with “C:\Program Files\Java\jdk1.8.0_172” in system variables.
- Download tomcat from “<https://tomcat.apache.org/download-90.cgi>”. Click on “zip” in “Core”. You will get a file called “apache-tomcat-9.0.7.zip”. Right click on “apache-tomcat-9.0.7.zip” and click on “Extract All”. Copy all contents of the extracted folder into “c:\tomcat” folder.
- Open Command Prompt and enter the following commands:
cd c:\tomcat\bin
startup.bat

c:\tomcat\webapps\ROOT\WEB-INF\classes\SampleServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

public class SampleServlet extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException,IOException
    {
        String str = request.getParameter("searchstr");
        PrintWriter out = response.getWriter();
        ArrayList<Employee> emps = new ArrayList<Employee>();
        emps.add(new Employee(1, "Scott", 4000));
        emps.add(new Employee(2, "Allen", 7500));
        emps.add(new Employee(3, "Jones", 9200));
        emps.add(new Employee(4, "James", 8400));
        emps.add(new Employee(5, "Smith", 5600));
        ArrayList<Employee> emps2 = new ArrayList<Employee>();
        for (int i = 0; i < emps.size(); i++)
        {
            if (emps.get(i).empname.indexOf(str) >= 0)
            {
                emps2.add(emps.get(i));
            }
        }
        String s = "[ ";
        for (int i = 0; i < emps2.size(); i++)
        {
            s += "{ \"empid\": \"" + emps2.get(i).empid + "\", \"empname\": \"" + emps2.get(i).empname + "\", "
            " \"salary\": \"" + emps2.get(i).salary + "\" }";
        }
    }
}
```

```
        if (i < emps2.size() - 1)
        {
            s += ",";
        }
    }
    s += "]";
    out.println(s);
}
}

class Employee
{
    public int empid;
    public String empname;
    public int salary;

    public Employee(int empid, String empname, int salary)
    {
        this.empid = empid;
        this.empname = empname;
        this.salary = salary;
    }
}
```

c:\tomcat\webapps\ROOT\WEB-INF\web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
    version="4.0"
    metadata-complete="true">

    <display-name>Welcome to Tomcat</display-name>
    <description>
        Welcome to Tomcat
    </description>

    <servlet>
        <servlet-name>SampleServlet</servlet-name>
        <servlet-class>SampleServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>SampleServlet</servlet-name>
        <url-pattern>/SampleServlet</url-pattern>
    </servlet-mapping>

</web-app>
```

Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular  
ng new app1  
cd c:\angular\app1  
ng g class Employee
```

c:\angular\app1\package.json

```
{  
  "name": "app1",  
  "version": "0.0.0",  
  "license": "MIT",  
  "scripts": {  
    "ng": "ng",  
    "start": "ng serve",  
    "build": "ng build --prod",  
    "test": "ng test",  
    "lint": "ng lint",  
    "e2e": "ng e2e"  
  },  
  "private": true,  
  "dependencies": {  
    "@angular/animations": "^5.2.0",  
    "@angular/common": "^5.2.0",  
    "@angular/compiler": "^5.2.0",  
    "@angular/core": "^5.2.0",  
    "@angular/forms": "^5.2.0",  
    "@angular/http": "^5.2.0",  
    "@angular/platform-browser": "^5.2.0",  
    "@angular/platform-browser-dynamic": "^5.2.0",  
    "@angular/router": "^5.2.0",  
    "core-js": "^2.4.1",  
    "rxjs": "^5.5.6",  
    "zone.js": "^0.8.19"  
  },  
  "devDependencies": {  
    "@angular/cli": "~1.7.4",  
    "@angular/compiler-cli": "^5.2.0",  
    "@angular/language-service": "^5.2.0",  
    "@types/jasmine": "~2.8.3",  
    "@types/jasminewd2": "~2.0.2",  
    "@types/node": "~6.0.60",  
    "codelyzer": "^4.0.1",  
    "jasmine-core": "~2.8.0",  
    "jasmine-spec-reporter": "~4.2.1",  
    "karma": "~2.0.0",  
    "karma-chrome-launcher": "~2.2.0",  
    "karma-coverage-istanbul-reporter": "^1.2.1",  
    "karma-jasmine": "~1.1.0",  
  }  
}
```

```
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\employee.ts

```
export class Employee
{
  empid: number;
  empname: string;
  salary: number;

  constructor(a, b, c)
  {
    this.empid = a;
    this.empname = b;
    this.salary = c;
  }
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from "@angular/forms";
import { HttpClientModule } from "@angular/common/http";

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

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule, FormsModule, HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

c:\angular\app1\src\app\app.component.ts

```
import { Component, Inject } from '@angular/core';
import { HttpClient } from "@angular/common/http";
import { Employee } from "./employee";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',

```

```

    styleUrls: ['./app.component.css']
  })
export class AppComponent
{
  s: string;
  employees: Employee[ ] = [];

  constructor( @Inject(HttpClient) private http: HttpClient)
  {
  }

  onSearchClick()
  {
    this.http.get<Employee>("/SampleServlet?searchstr=" + this.s, { responseType: "json"
  }).subscribe(this.onAjaxSuccess, this.onAjaxError);
  }

  onAjaxSuccess = (response) =>
  {
    this.employees = response;
  }

  onAjaxError = () =>
  {
    alert("error");
  }
}

```

c:\angular\app1\src\app\app.component.html

```

<div>
  <h4>Ajax - Java - Search</h4>
  <input type="text" [(ngModel)]="s">
  <input type="button" value="Search" (click)="onSearchClick()">
  <table border="1">
    <tr>
      <th>Emp ID</th>
      <th>Emp Name</th>
      <th>Salary</th>
    </tr>
    <tr *ngFor="let employee of employees">
      <td>{{employee.empid}}</td>
      <td>{{employee.empname}}</td>
      <td>{{employee.salary}}</td>
    </tr>
  </table>
</div>

```

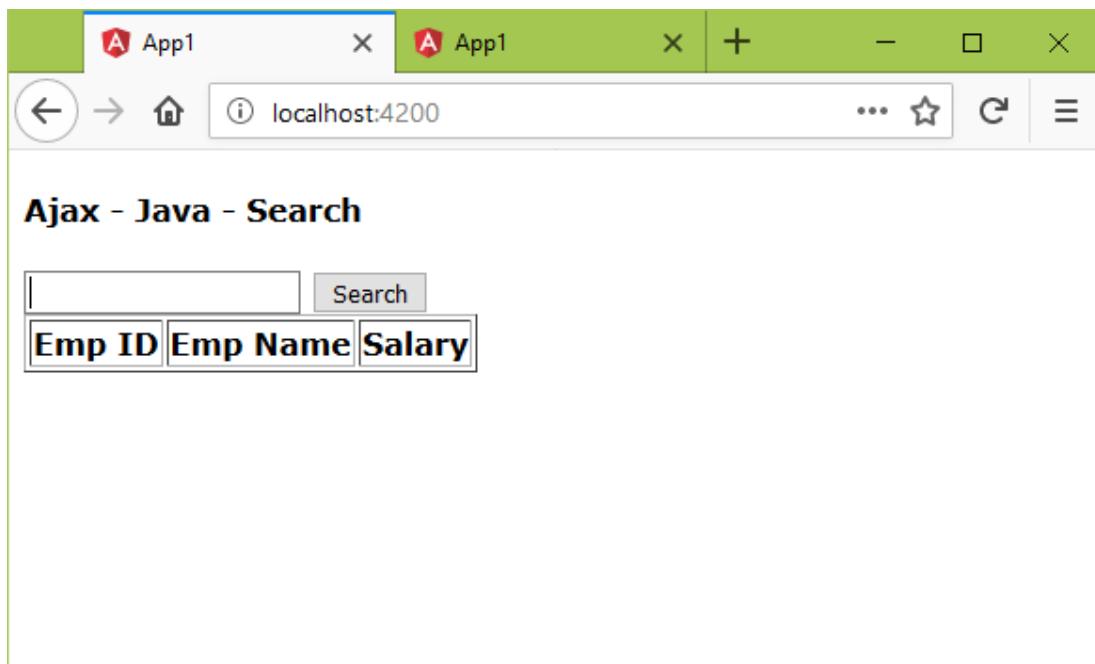
Executing the application:

- Open Command Prompt and enter the following commands:

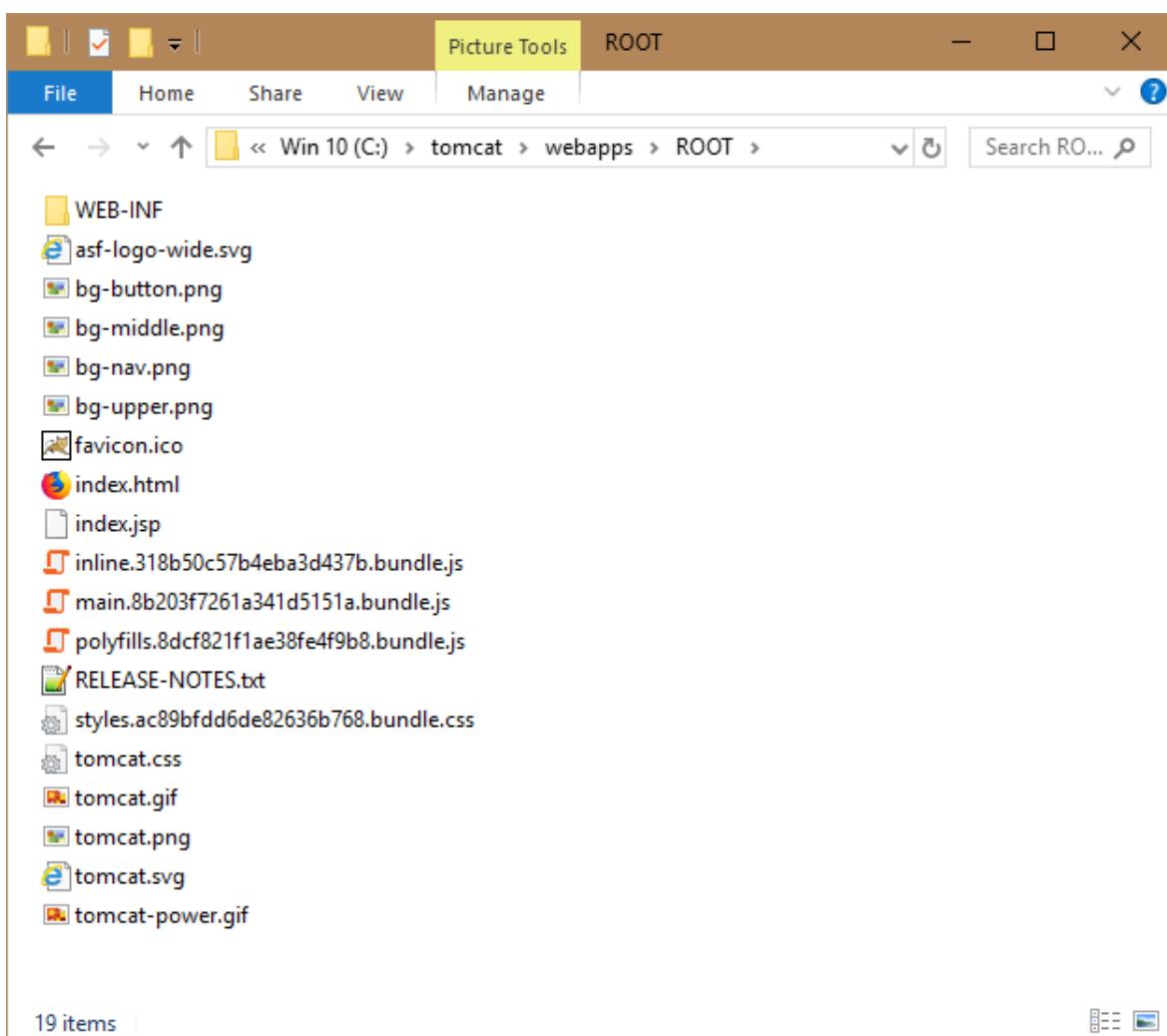
```
cd c:\tomcat\webapps\ROOT\WEB-INF\classes  
javac -cp c:\tomcat\lib\servlet-api.jar SampleServlet.java  
cd c:\angular\app1  
ng build --prod  
ng serve
```

- Open the browser and enter the following URL:

<http://localhost:4200>

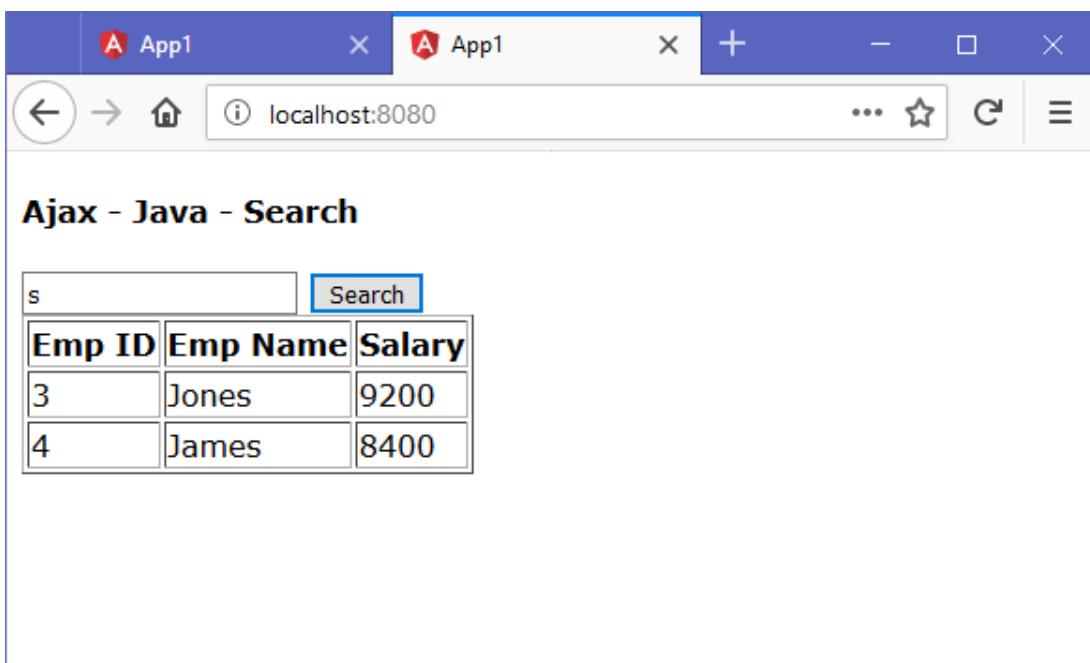


- Copy all files from "c:\angular\app1\dist" folder to "c:\tomcat\webapps\ROOT".



- Open the browser and enter the following URL:

<http://localhost:8080/index.html>



Type some text in the search textbox and click on “Search”.

AJAX - Java - Insert - Example

Setting-up Environment for Java

- Install Java from “<https://java.com/en/download>”.
- Add “C:\Program Files\Java\jdk1.8.0_172\bin” as “Path” of system variables.
- Add “JAVA_HOME” with “C:\Program Files\Java\jdk1.8.0_172” in system variables.
- Download tomcat from “<https://tomcat.apache.org/download-90.cgi>”. Click on “zip” in “Core”. You will get a file called “apache-tomcat-9.0.7.zip”. Right click on “apache-tomcat-9.0.7.zip” and click on “Extract All”. Copy all contents of the extracted folder into “c:\tomcat” folder.
- Open Command Prompt and enter the following commands:
cd c:\tomcat\bin
startup.bat

c:\tomcat\webapps\ROOT\WEB-INF\classes\SampleServlet.java

```

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

public class SampleServlet extends HttpServlet
{
    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException
    {
        String a = request.getParameter("empid");
    }
}

```

```

        String b = request.getParameter("empname");
        String c = request.getParameter("salary");
        //write code for inserting
        PrintWriter out = response.getWriter();
        out.println("Successfully Inserted");
    }
}

```

c:\tomcat\webapps\ROOT\WEB-INF\web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
  version="4.0"
  metadata-complete="true">

  <display-name>Welcome to Tomcat</display-name>
  <description>
    Welcome to Tomcat
  </description>

  <servlet>
    <servlet-name>SampleServlet</servlet-name>
    <servlet-class>SampleServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>SampleServlet</servlet-name>
    <url-pattern>/SampleServlet</url-pattern>
  </servlet-mapping>

</web-app>

```

Creating Application

- Open Command Prompt and enter the following commands:

```

cd c:\angular
ng new app1
cd c:\angular\app1
ng g class Employee

```

c:\angular\app1\package.json

```
{
  "name": "app1",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "prestart": "tsc"
  }
}
```

```
"build": "ng build --prod",
"test": "ng test",
"lint": "ng lint",
"e2e": "ng e2e"
},
"private": true,
"dependencies": {
  "@angular/animations": "^5.2.0",
  "@angular/common": "^5.2.0",
  "@angular/compiler": "^5.2.0",
  "@angular/core": "^5.2.0",
  "@angular/forms": "^5.2.0",
  "@angular/http": "^5.2.0",
  "@angular/platform-browser": "^5.2.0",
  "@angular/platform-browser-dynamic": "^5.2.0",
  "@angular/router": "^5.2.0",
  "core-js": "^2.4.1",
  "rxjs": "^5.5.6",
  "zone.js": "^0.8.19"
},
"devDependencies": {
  "@angular/cli": "~1.7.4",
  "@angular/compiler-cli": "^5.2.0",
  "@angular/language-service": "^5.2.0",
  "@types/jasmine": "~2.8.3",
  "@types/jasminewd2": "~2.0.2",
  "@types/node": "~6.0.60",
  "codelyzer": "^4.0.1",
  "jasmine-core": "~2.8.0",
  "jasmine-spec-reporter": "~4.2.1",
  "karma": "~2.0.0",
  "karma-chrome-launcher": "~2.2.0",
  "karma-coverage-istanbul-reporter": "^1.2.1",
  "karma-jasmine": "~1.1.0",
  "karma-jasmine-html-reporter": "^0.2.2",
  "protractor": "~5.1.2",
  "ts-node": "~4.1.0",
  "tslint": "~5.9.1",
  "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\employee.ts

```
export class Employee
{
  empid: number;
  empname: string;
  salary: number;

  constructor(a, b, c)
  {
    this.empid = a;
```

```
this.empname = b;  
this.salary = c;  
}  
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';  
import { NgModule } from '@angular/core';  
import { FormsModule } from "@angular/forms";  
import { HttpClientModule } from "@angular/common/http";  
  
import { AppComponent } from './app.component';  
  
@NgModule({  
  declarations: [  
    AppComponent  
  ],  
  imports: [  
    BrowserModule, FormsModule, HttpClientModule  
  ],  
  providers: [],  
  bootstrap: [AppComponent]  
})  
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component, Inject } from '@angular/core';  
import { HttpClient } from "@angular/common/http";  
import { Employee } from "./employee";  
  
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})  
export class AppComponent  
{  
  msg: string;  
  emp: Employee = new Employee(null, null, null);  
  
  constructor( @Inject(HttpClient) private http: HttpClient)  
  {}  
  
  onInsertClick()  
  {  
    this.http.post("/SampleServlet", this.emp, { responseType: "text" }).subscribe(this.onAjaxSuccess,  
    this.onAjaxError);  
  }  
  
  onAjaxSuccess = (response) =>
```

```
{  
  this.msg = response;  
}  
  
onAjaxError = (error) =>  
{  
  alert(error);  
}  
}
```

c:\angular\app1\src\app\app.component.html

```
<div>  
  <h4>Ajax - Java - Insert</h4>  
  <form>  
    Emp ID: <input type="text" [(ngModel)]="emp.empid" name="empid"><br>  
    Emp Name: <input type="text" [(ngModel)]="emp.empname" name="empname"><br>  
    Salary: <input type="text" [(ngModel)]="emp.salary" name="salary" /><br>  
    <input type="submit" value="Insert" (click)="onInsertClick()" /><br>  
    {{msg}}  
  </form>  
</div>
```

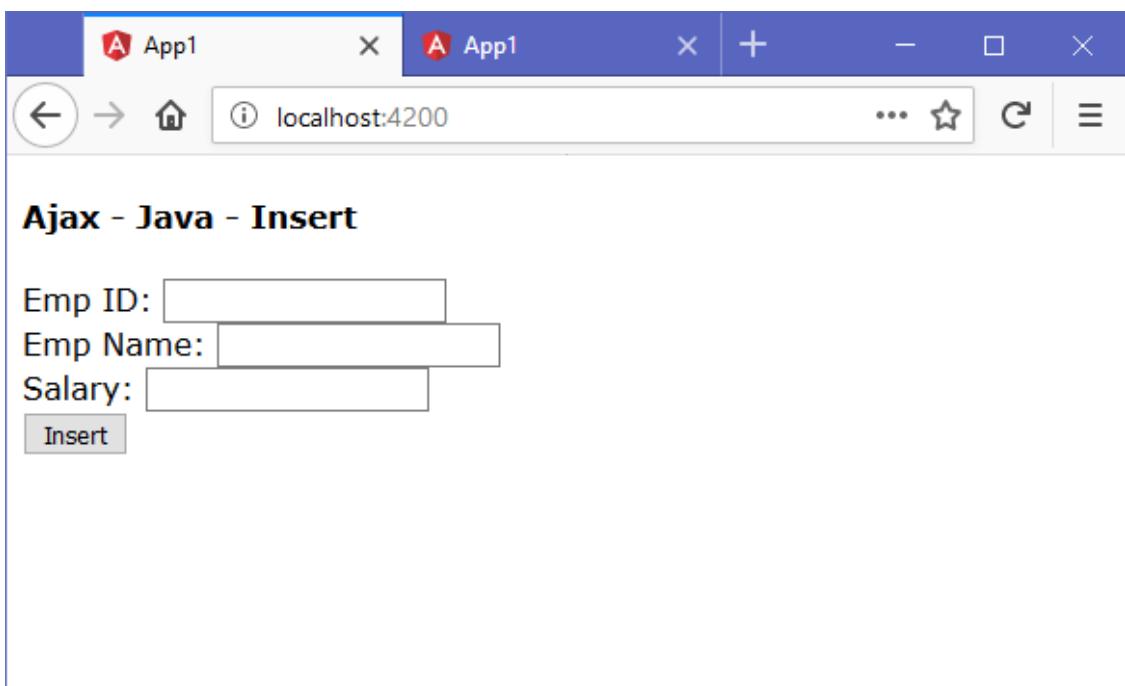
Executing the application:

- Open Command Prompt and enter the following commands:

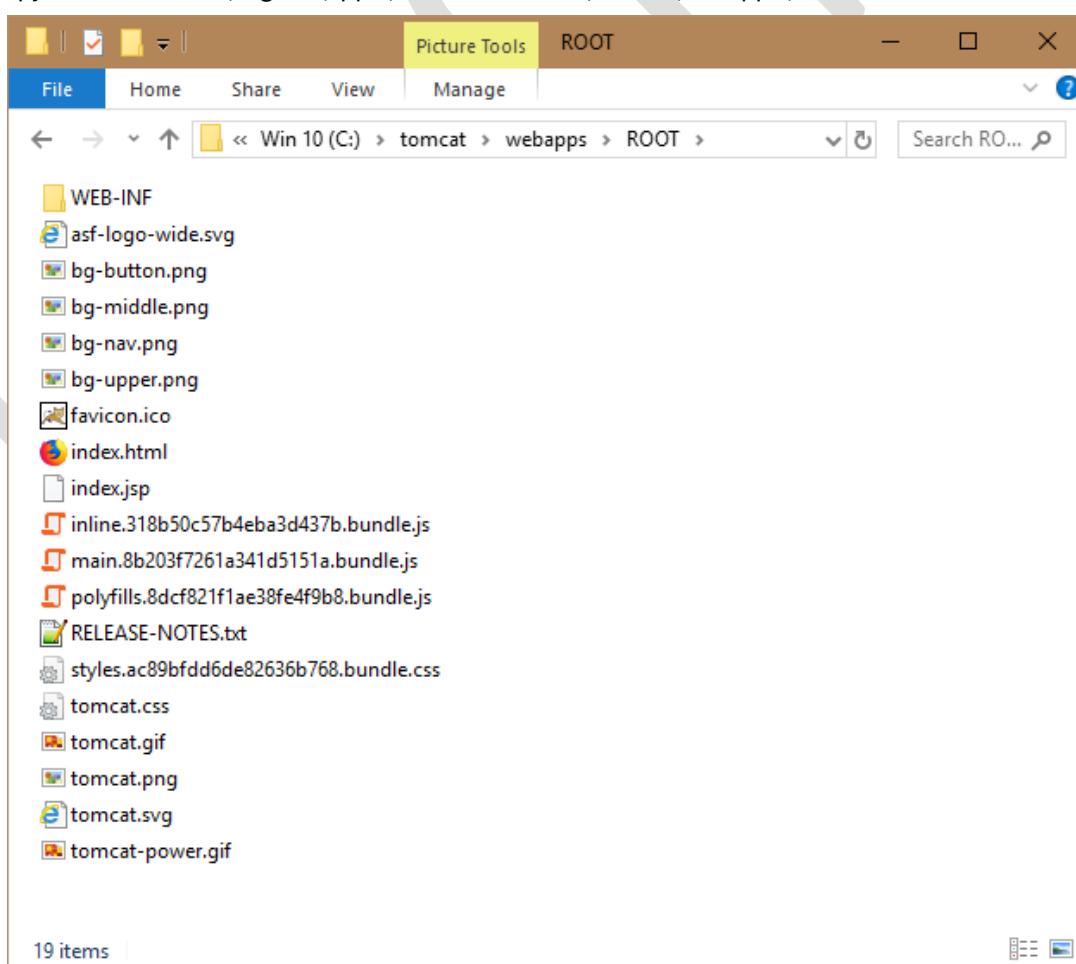
```
cd c:\tomcat\webapps\ROOT\WEB-INF\classes  
javac -cp c:\tomcat\lib\servlet-api.jar SampleServlet.java  
cd c:\angular\app1  
ng build --prod  
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```

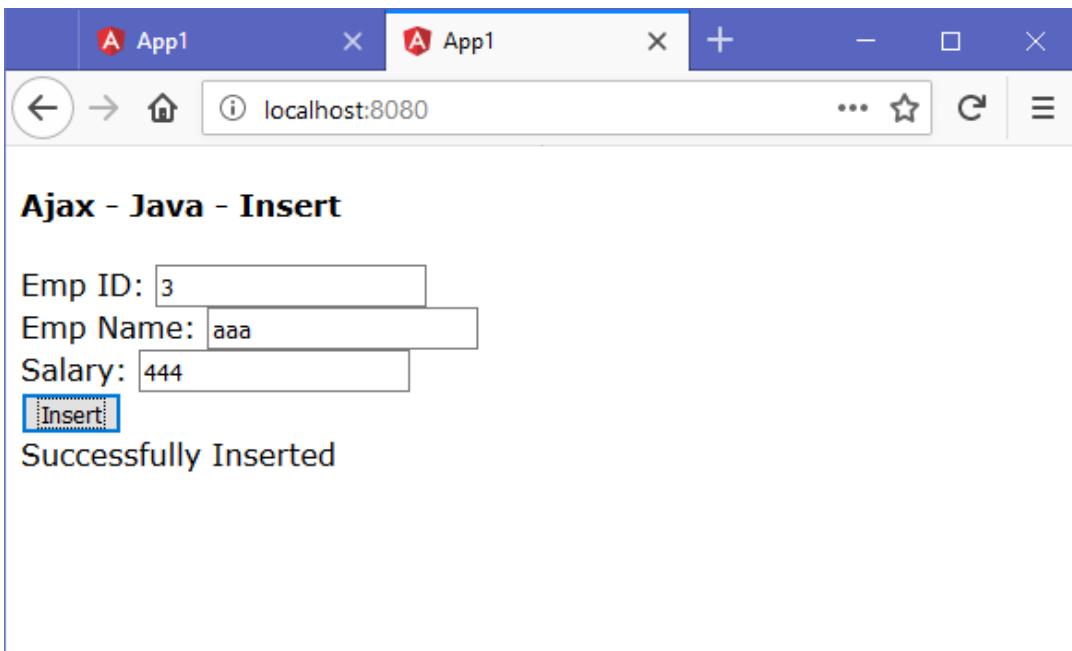


- Copy all files from "c:\angular\app1\dist" folder to "c:\tomcat\webapps\ROOT".



- Open the browser and enter the following URL:

<http://localhost:8080/index.html>



Type some empid, empname and salary and then click on “Insert”.

AJAX – Java – Update - Example

Setting-up Environment for Java

- Install Java from “<https://java.com/en/download>”.
- Add “C:\Program Files\Java\jdk1.8.0_172\bin” as “Path” of system variables.
- Add “JAVA_HOME” with “C:\Program Files\Java\jdk1.8.0_172” in system variables.
- Download tomcat from “<https://tomcat.apache.org/download-90.cgi>”. Click on “zip” in “Core”. You will get a file called “apache-tomcat-9.0.7.zip”. Right click on “apache-tomcat-9.0.7.zip” and click on “Extract All”. Copy all contents of the extracted folder into “c:\tomcat” folder.
- Open Command Prompt and enter the following commands:
cd c:\tomcat\bin
startup.bat

c:\tomcat\webapps\ROOT\WEB-INF\classes\SampleServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

public class SampleServlet extends HttpServlet
{
    public void doPut(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException
    {
    }
```

```

        String a = request.getParameter("empid");
        String b = request.getParameter("empname");
        String c = request.getParameter("salary");
        //write code for updating
        PrintWriter out = response.getWriter();
        out.println("Successfully Updated");
    }
}

```

c:\tomcat\webapps\ROOT\WEB-INF\web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
  version="4.0"
  metadata-complete="true">

  <display-name>Welcome to Tomcat</display-name>
  <description>
    Welcome to Tomcat
  </description>

  <servlet>
    <servlet-name>SampleServlet</servlet-name>
    <servlet-class>SampleServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>SampleServlet</servlet-name>
    <url-pattern>/SampleServlet</url-pattern>
  </servlet-mapping>

</web-app>

```

Creating Application

- Open Command Prompt and enter the following commands:

```

cd c:\angular
ng new app1
cd c:\angular\app1
ng g class Employee

```

c:\angular\app1\package.json

```
{
  "name": "app1",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
  }
}
```

```
"start": "ng serve",
"build": "ng build --prod",
"test": "ng test",
"lint": "ng lint",
"e2e": "ng e2e"
},
"private": true,
"dependencies": {
  "@angular/animations": "^5.2.0",
  "@angular/common": "^5.2.0",
  "@angular/compiler": "^5.2.0",
  "@angular/core": "^5.2.0",
  "@angular/forms": "^5.2.0",
  "@angular/http": "^5.2.0",
  "@angular/platform-browser": "^5.2.0",
  "@angular/platform-browser-dynamic": "^5.2.0",
  "@angular/router": "^5.2.0",
  "core-js": "^2.4.1",
  "rxjs": "^5.5.6",
  "zone.js": "^0.8.19"
},
"devDependencies": {
  "@angular/cli": "~1.7.4",
  "@angular/compiler-cli": "^5.2.0",
  "@angular/language-service": "^5.2.0",
  "@types/jasmine": "~2.8.3",
  "@types/jasminewd2": "~2.0.2",
  "@types/node": "~6.0.60",
  "codelyzer": "^4.0.1",
  "jasmine-core": "~2.8.0",
  "jasmine-spec-reporter": "~4.2.1",
  "karma": "~2.0.0",
  "karma-chrome-launcher": "~2.2.0",
  "karma-coverage-istanbul-reporter": "^1.2.1",
  "karma-jasmine": "~1.1.0",
  "karma-jasmine-html-reporter": "^0.2.2",
  "protractor": "~5.1.2",
  "ts-node": "~4.1.0",
  "tslint": "~5.9.1",
  "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\employee.ts

```
export class Employee
{
  empid: number;
  empname: string;
  salary: number;

  constructor(a, b, c)
  {
```

```
this.empid = a;  
this.empname = b;  
this.salary = c;  
}  
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';  
import { NgModule } from '@angular/core';  
import { FormsModule } from "@angular/forms";  
import { HttpClientModule } from "@angular/common/http";  
  
import { AppComponent } from './app.component';  
  
@NgModule({  
  declarations: [  
    AppComponent  
  ],  
  imports: [  
    BrowserModule, FormsModule, HttpClientModule  
  ],  
  providers: [],  
  bootstrap: [AppComponent]  
})  
export class AppModule { }
```

c:\angular\app1\src\app\app.component.ts

```
import { Component, Inject } from '@angular/core';  
import { HttpClient } from "@angular/common/http";  
import { Employee } from "./employee";  
  
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})  
export class AppComponent  
{  
  msg: string;  
  emp: Employee = new Employee(null, null, null);  
  
  constructor( @Inject(HttpClient) private http: HttpClient)  
  {}  
  
  onUpdateClick()  
  {  
    this.http.put("/SampleServlet", this.emp, { responseType: "text" }).subscribe(this.onAjaxSuccess,  
    this.onAjaxError);  
  }  
}
```

```
onAjaxSuccess = (response) =>
{
  this.msg = response;
}

onAjaxError = (error) =>
{
  alert(error);
}
```

c:\angular\app1\src\app\app.component.html

```
<div>
<h4>Ajax - Java - Update</h4>
<form>
  Existing Emp ID: <input type="text" [(ngModel)]="emp.empid" name="empid"><br>
  Emp Name: <input type="text" [(ngModel)]="emp.empname" name="empname"><br>
  Salary: <input type="text" [(ngModel)]="emp.salary" name="salary" /><br>
  <input type="submit" value="Update" (click)="onUpdateClick()" /><br>
  {{msg}}
</form>
</div>
```

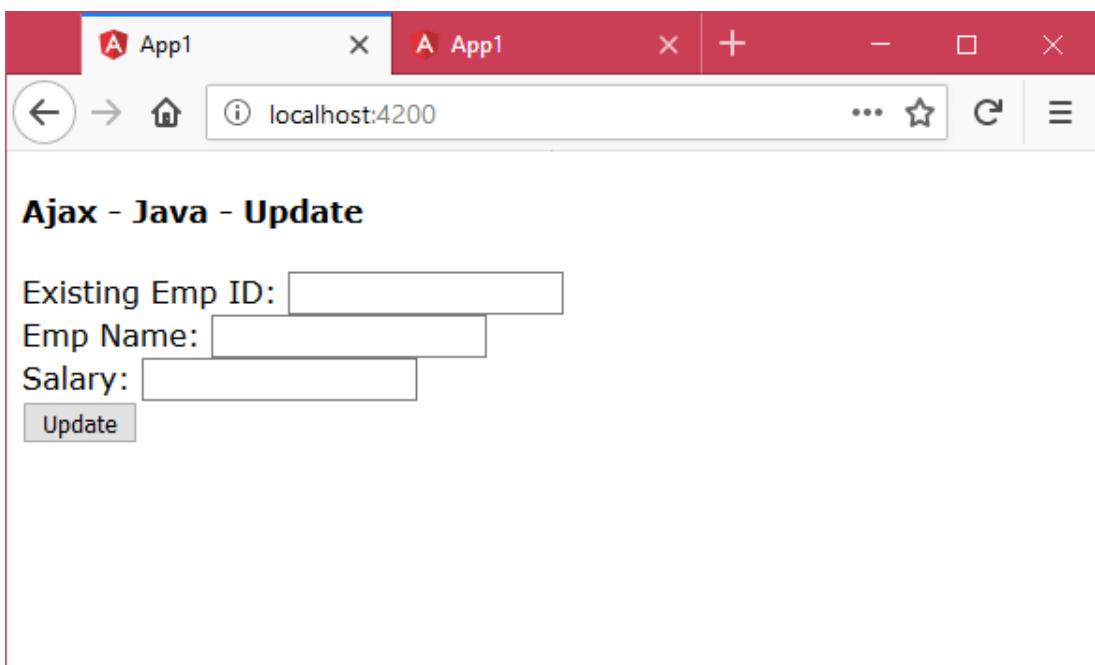
Executing the application:

- Open Command Prompt and enter the following commands:

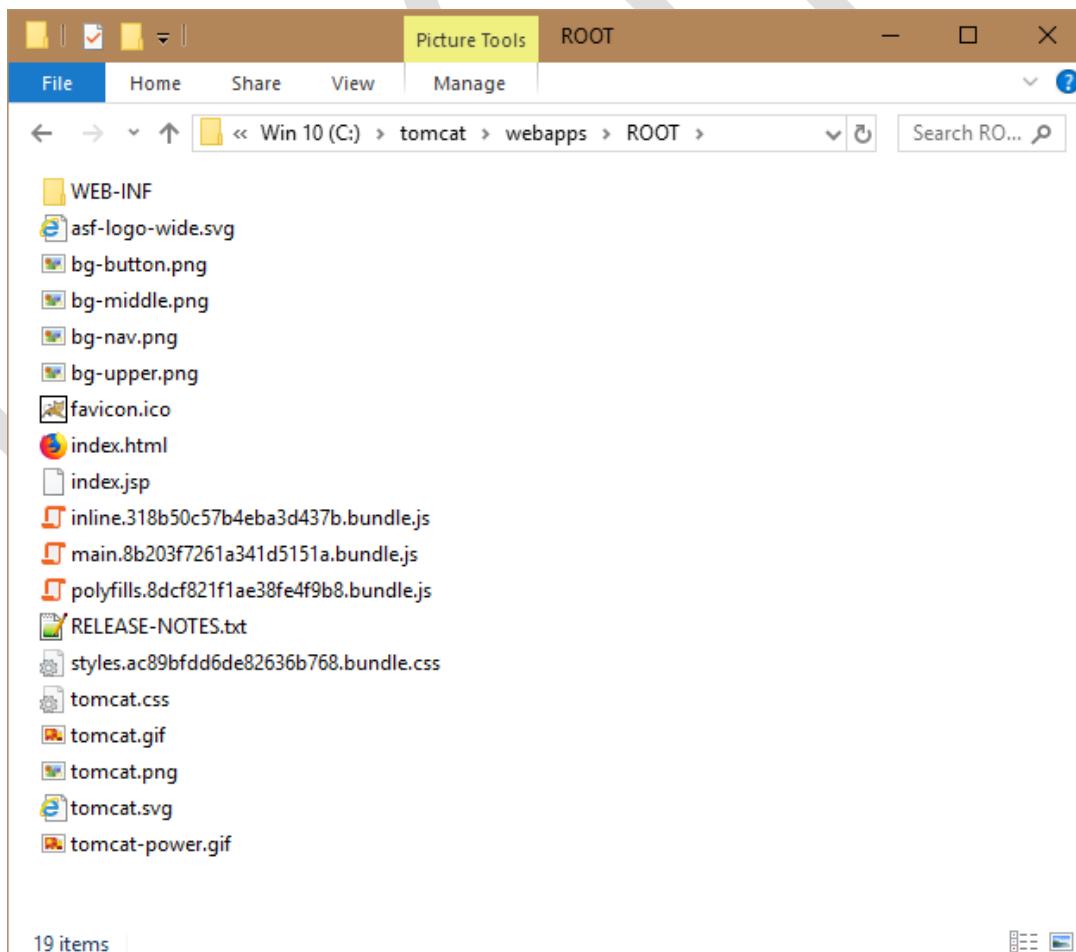
```
cd c:\tomcat\webapps\ROOT\WEB-INF\classes
javac -cp c:\tomcat\lib\servlet-api.jar SampleServlet.java
cd c:\angular\app1
ng build --prod
ng serve
```

- Open the browser and enter the following URL:

<http://localhost:4200>

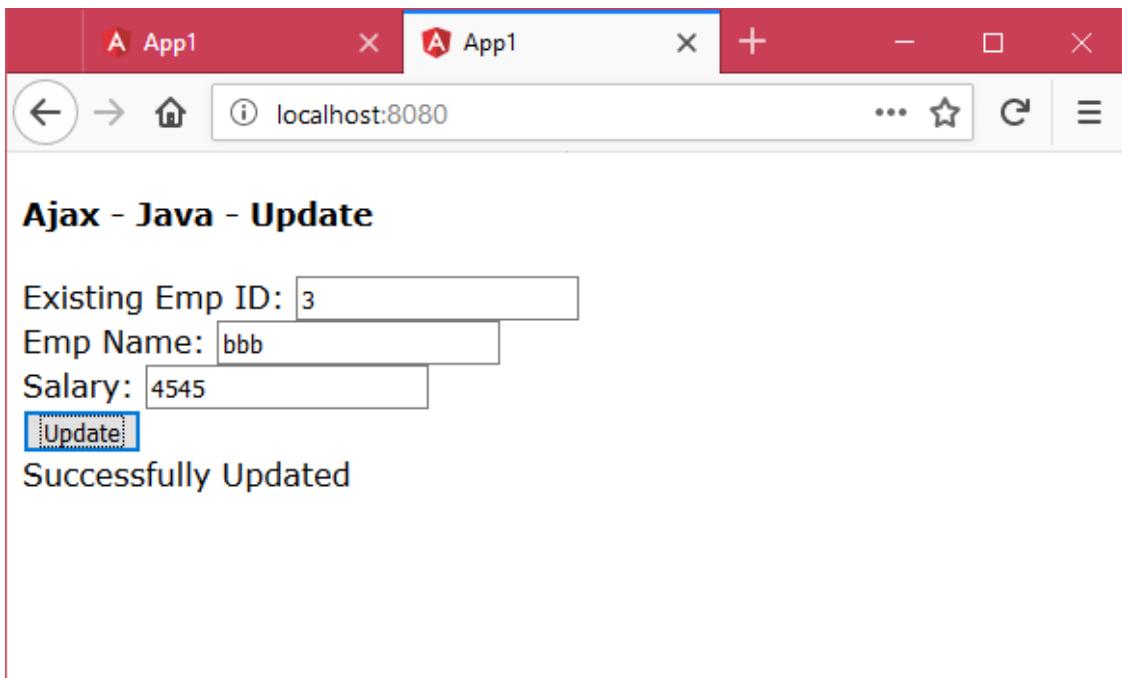


- Copy all files from “c:\angular\app1\dist” folder to “c:\tomcat\webapps\ROOT”.



- Open the browser and enter the following URL:

<http://localhost:8080/index.html>



Type some empid, empname and salary and then click on “Update”.

AJAX – Java – Delete - Example

Setting-up Environment for Java

- Install Java from “<https://java.com/en/download>”.
- Add “C:\Program Files\Java\jdk1.8.0_172\bin” as “Path” of system variables.
- Add “JAVA_HOME” with “C:\Program Files\Java\jdk1.8.0_172” in system variables.
- Download tomcat from “<https://tomcat.apache.org/download-90.cgi>”. Click on “zip” in “Core”. You will get a file called “apache-tomcat-9.0.7.zip”. Right click on “apache-tomcat-9.0.7.zip” and click on “Extract All”. Copy all contents of the extracted folder into “c:\tomcat” folder.
- Open Command Prompt and enter the following commands:
cd c:\tomcat\bin
startup.bat

c:\tomcat\webapps\ROOT\WEB-INF\classes\SampleServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

public class SampleServlet extends HttpServlet
{
    public void doDelete(HttpServletRequest request, HttpServletResponse response) throws
ServletException,IOException
```

```
{
    String a = request.getParameter("empid");
    //write code for deleting
    PrintWriter out = response.getWriter();
    out.println("Successfully Deleted");
}
}
```

c:\tomcat\webapps\ROOT\WEB-INF\web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
  version="4.0"
  metadata-complete="true">

  <display-name>Welcome to Tomcat</display-name>
  <description>
    Welcome to Tomcat
  </description>

  <servlet>
    <servlet-name>SampleServlet</servlet-name>
    <servlet-class>SampleServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>SampleServlet</servlet-name>
    <url-pattern>/SampleServlet</url-pattern>
  </servlet-mapping>

</web-app>
```

Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
```

```
ng new app1
```

c:\angular\app1\package.json

```
{
  "name": "app1",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build --prod",
    "test": "ng test",
    "lint": "ng lint",
```

```

    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^5.2.0",
    "@angular/common": "^5.2.0",
    "@angular/compiler": "^5.2.0",
    "@angular/core": "^5.2.0",
    "@angular/forms": "^5.2.0",
    "@angular/http": "^5.2.0",
    "@angular/platform-browser": "^5.2.0",
    "@angular/platform-browser-dynamic": "^5.2.0",
    "@angular/router": "^5.2.0",
    "core-js": "^2.4.1",
    "rxjs": "^5.5.6",
    "zone.js": "^0.8.19"
  },
  "devDependencies": {
    "@angular/cli": "~1.7.4",
    "@angular/compiler-cli": "^5.2.0",
    "@angular/language-service": "^5.2.0",
    "@types/jasmine": "~2.8.3",
    "@types/jasminewd2": "~2.0.2",
    "@types/node": "~6.0.60",
    "codelyzer": "^4.0.1",
    "jasmine-core": "~2.8.0",
    "jasmine-spec-reporter": "~4.2.1",
    "karma": "~2.0.0",
    "karma-chrome-launcher": "~2.2.0",
    "karma-coverage-istanbul-reporter": "^1.2.1",
    "karma-jasmine": "~1.1.0",
    "karma-jasmine-html-reporter": "^0.2.2",
    "protractor": "~5.1.2",
    "ts-node": "~4.1.0",
    "tslint": "~5.9.1",
    "typescript": "~2.5.3"
  }
}

```

c:\angular\app1\src\app\app.module.ts

```

import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';

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

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [

```

```
BrowserModule, FormsModule, HttpClientModule
],
providers: [],
bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component, Inject } from '@angular/core';
import { HttpClient } from "@angular/common/http";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{
  msg: string;
  empid: number;

  constructor( @Inject(HttpClient) private http: HttpClient)
  {

  }

  onDeleteClick()
  {
    this.http.delete("/SampleServlet?empid=" + this.empid, { responseType: "text"
  }).subscribe(this.onAjaxSuccess, this.onAjaxError);
  }

  onAjaxSuccess = (response) =>
  {
    this.msg = response;
  }

  onAjaxError = (error) =>
  {
    alert(error);
  }
}
```

c:\angular\app1\src\app\app.component.html

```
<div>
<h4>Ajax - Java - Delete</h4>
<form>
  Existing Emp ID: <input type="text" [(ngModel)]="empid" name="empid"><br>
  <input type="submit" value="Delete" (click)="onDeleteClick()" /><br>
  {{msg}}
</form>
</div>
```

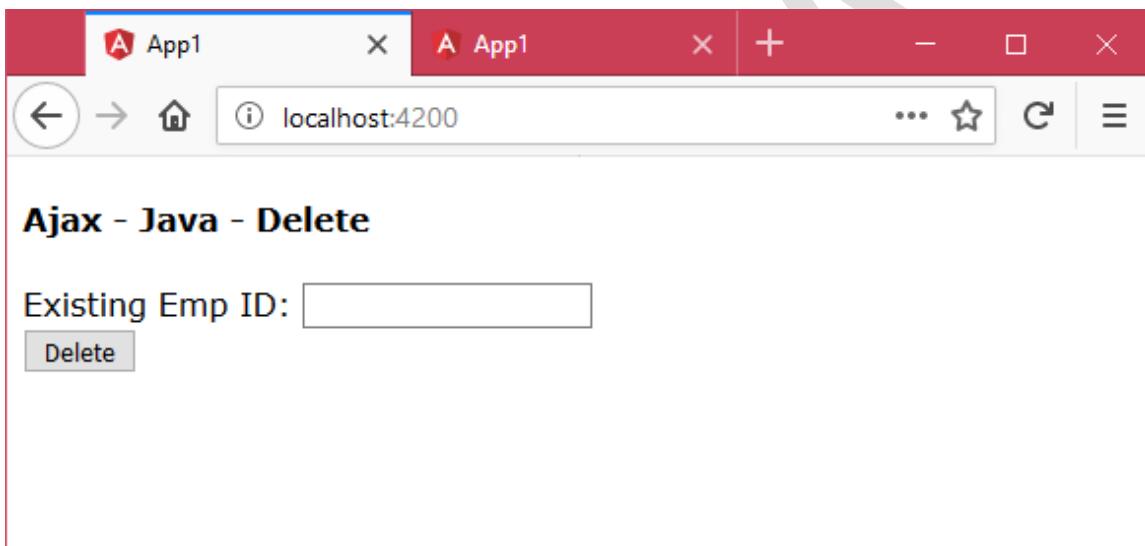
Executing the application:

- Open Command Prompt and enter the following commands:

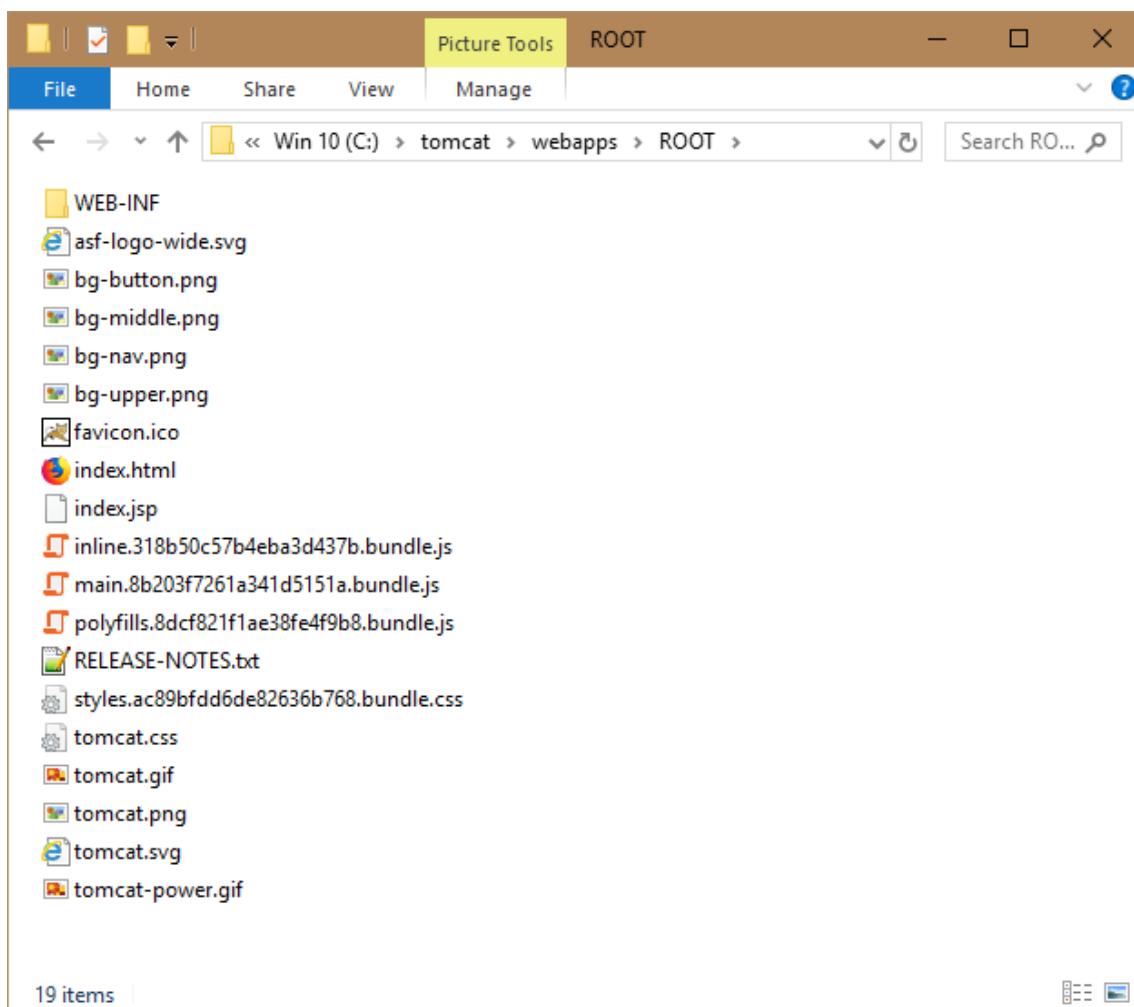
```
cd c:\tomcat\webapps\ROOT\WEB-INF\classes  
javac -cp c:\tomcat\lib\servlet-api.jar SampleServlet.java  
cd c:\angular\app1  
ng build --prod  
ng serve
```

- Open the browser and enter the following URL:

<http://localhost:4200>

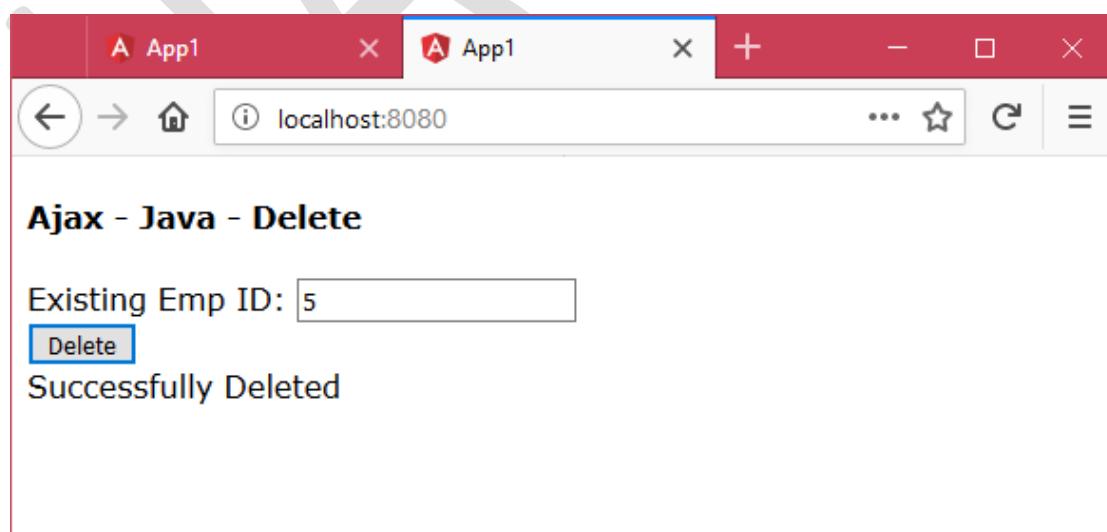


- Copy all files from "c:\angular\app1\dist" folder to "c:\tomcat\webapps\ROOT".



- Open the browser and enter the following URL:

<http://localhost:8080/index.html>



Http Interceptors

- Http Interceptor is a angular service, that adds one or more requests headers automatically for each ajax request sent from the application.
- These are used to implement authentication.
- Authentication is a process of checking whether the user is a valid user or not.
- First, the server generates a random number, stores it at server and also sends it to the client as response. The client (browser) stores it in the localStorage. Next, the client sends the same random number to the server via HTTP header, by using Http Interceptor. Then the server verifies whether the received random number is matching with the stored random number. If matching, it is a valid user; otherwise invalid user. Thus authentication is performed.
- Http Interceptors added as a provider in the module level; automatically executes for all ajax requests sent via HttpClient within the same module and its child modules.
- Http Interceptor is a service class that implements HttpInterceptor interface. This interface enforces that the service class must contain a method called “intercept” with two parameters “request” or HttpRequest type and “next” of HttpHandler type and returns HttpEvent type.
- The “intercept” method automatically executes before an AJAX request (Http Request) is sent to browser.
- The “request” argument represents the current request. The “next” argument represents the next interceptor.
- The “HttpEvent” represents each single process that happens before sending http request to server.
- The “request.clone” method is used to copy the request into a new request object; so that we can add headers to the request.
- The next.handle() method calls the next interceptor, if any.

Steps for Working with Http Interceptor

- **Create Service:**

```

import { Injectable } from "@angular/core";
import { HttpInterceptor, HttpRequest, HttpHandler, HttpEvent } from "@angular/common/http";
import { Observable } from "rxjs/Observable";

@Injectable()
export class Serviceclassname implements HttpInterceptor
{
  intercept(request, next: HttpHandler): Observable<HttpEvent<any>>
  {
    request = request.clone({ setHeaders: { "mykey": "myvaluue" } });
    return next.handle(request);
  }
}

```

```
}
```

- **Add Http Interceptor Service as provider in the component:**

```
import { HTTP_INTERCEPTORS } from "@angular/common/http";
@Component( { ..., providers: [ { provide: HTTP_INTERCEPTORS, useClass: SampleInterceptorService, multi: true } ] })
export class Moduleclassname
{
```

AJAX – Java – Http Interceptors - Get - Example

Setting-up Environment for Java

- Install Java from “<https://java.com/en/download>”.
- Add “C:\Program Files\Java\jdk1.8.0_172\bin” as “Path” of system variables.
- Add “JAVA_HOME” with “C:\Program Files\Java\jdk1.8.0_172” in system variables.
- Download tomcat from “<https://tomcat.apache.org/download-90.cgi>”. Click on “zip” in “Core”. You will get a file called “apache-tomcat-9.0.7.zip”. Right click on “apache-tomcat-9.0.7.zip” and click on “Extract All”. Copy all contents of the extracted folder into “c:\tomcat” folder.
- Open Command Prompt and enter the following commands:
cd c:\tomcat\bin
startup.bat

c:\tomcat\webapps\ROOT\WEB-INF\classes\SampleServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SampleServlet extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
    {
        PrintWriter out = response.getWriter();
        if (request.getHeader("mykey").equals("myvalue"))
        {
            out.println("[ { \"empid\": 1, \"empname\": \"Scott\", \"salary\": 4000 }, { \"empid\": 2,
\"empname\": \"Allen\", \"salary\": 7500 }, { \"empid\": 3, \"empname\": \"Jones\", \"salary\": 9200 }, {
\"empid\": 4, \"empname\": \"James\", \"salary\": 8400 }, { \"empid\": 5, \"empname\": \"Smith\", \"salary\": 5600 } ]");
        }
        else
        {
```

```

        out.println("[ { \"empid\": 0, \"empname\": \"no data\", \"salary\": 0 } ]");
    }
}
}

```

c:\tomcat\webapps\ROOT\WEB-INF\web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
  version="4.0"
  metadata-complete="true">

  <display-name>Welcome to Tomcat</display-name>
  <description>
    Welcome to Tomcat
  </description>

  <servlet>
    <servlet-name>SampleServlet</servlet-name>
    <servlet-class>SampleServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>SampleServlet</servlet-name>
    <url-pattern>/SampleServlet</url-pattern>
  </servlet-mapping>

</web-app>

```

Creating Application

- Open Command Prompt and enter the following commands:

```

cd c:\angular
ng new app1
cd c:\angular\app1
ng g class Employee
ng g service Employees
ng g service SampleInterceptor

```

c:\angular\app1\package.json

```
{
  "name": "app1",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
  }
}
```

```
"start": "ng serve",
"build": "ng build --prod",
"test": "ng test",
"lint": "ng lint",
"e2e": "ng e2e"
},
"private": true,
"dependencies": {
  "@angular/animations": "^5.2.0",
  "@angular/common": "^5.2.0",
  "@angular/compiler": "^5.2.0",
  "@angular/core": "^5.2.0",
  "@angular/forms": "^5.2.0",
  "@angular/http": "^5.2.0",
  "@angular/platform-browser": "^5.2.0",
  "@angular/platform-browser-dynamic": "^5.2.0",
  "@angular/router": "^5.2.0",
  "core-js": "^2.4.1",
  "rxjs": "^5.5.6",
  "zone.js": "^0.8.19"
},
"devDependencies": {
  "@angular/cli": "~1.7.4",
  "@angular/compiler-cli": "^5.2.0",
  "@angular/language-service": "^5.2.0",
  "@types/jasmine": "~2.8.3",
  "@types/jasminewd2": "~2.0.2",
  "@types/node": "~6.0.60",
  "codelyzer": "^4.0.1",
  "jasmine-core": "~2.8.0",
  "jasmine-spec-reporter": "~4.2.1",
  "karma": "~2.0.0",
  "karma-chrome-launcher": "~2.2.0",
  "karma-coverage-istanbul-reporter": "^1.2.1",
  "karma-jasmine": "~1.1.0",
  "karma-jasmine-html-reporter": "^0.2.2",
  "protractor": "~5.1.2",
  "ts-node": "~4.1.0",
  "tslint": "~5.9.1",
  "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\employee.ts

```
export class Employee
{
  empid: number;
  empname: string;
  salary: number;

  constructor(a, b, c)
  {
```

```
this.empid = a;  
this.empname = b;  
this.salary = c;  
}  
}
```

c:\angular\app1\src\app\employees.service.ts

```
import { Injectable, Inject } from '@angular/core';  
import { Employee } from './employee';  
import { HttpClient } from '@angular/common/http';  
  
@Injectable()  
export class EmployeesService  
{  
  constructor( @Inject(HttpClient) private http: HttpClient)  
  {}  
  
  resolve: any;  
  reject: any;  
  
  getEmployees()  
  {  
    this.http.get<Employee>("/SampleServlet", { responseType: "json" }).subscribe(this.onAjaxSuccess,  
    this.onAjaxError);  
    return new Promise(this.promiseCallback);  
  }  
  
  promiseCallback = (resolve, reject) =>  
  {  
    this.resolve = resolve;  
    this.reject = reject;  
  }  
  
  onAjaxSuccess = (response) =>  
  {  
    this.resolve(response);  
  }  
  
  onAjaxError = () =>  
  {  
    this.reject("Failed");  
  }  
}
```

c:\angular\app1\src\app\sample-interceptor.service.ts

```
import { Injectable } from '@angular/core';  
import { HttpInterceptor, HttpRequest, HttpHandler, HttpEvent } from '@angular/common/http';  
import { Observable } from 'rxjs/Observable';  
  
@Injectable()
```

```
export class SampleInterceptorService implements HttpInterceptor
{
  intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>>
  {
    request = request.clone({ setHeaders: { "mykey": "myvalue" } });
    return next.handle(request);
  }
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from "@angular/forms";
import { HttpClientModule } from "@angular/common/http";

import { AppComponent } from './app.component';
import { EmployeesService } from './employees.service';
import { HTTP_INTERCEPTORS } from "@angular/common/http";
import { SampleInterceptorService } from "./sample-interceptor.service";

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule, FormsModule, HttpClientModule
  ],
  providers: [
    EmployeesService,
    { provide: HTTP_INTERCEPTORS, useClass: SampleInterceptorService, multi: true }
  ],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component, Inject } from '@angular/core';
import { EmployeesService } from "./employees.service";
import { Employee } from "./employee";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{
  employees: Employee[] = [];

  constructor( @Inject(EmployeesService) private employeesService: EmployeesService)
  {
```

```
}

onGetDataClick()
{
  this.employeesService.getEmployees().then(this.onPromiseSuccess, this.onPromiseError);
}

onPromiseSuccess = (response) =>
{
  this.employees = response;
}

onPromiseError = () =>
{
  alert("error");
}
```

c:\angular\app1\src\app\app.component.html

```
<div>
  <h4>Ajax - Http Interceptors - Promise - Java - Get</h4>
  <input type="button" value="Get Data" (click)="onGetDataClick()">
  <table border="1">
    <tr>
      <th>Emp ID</th>
      <th>Emp Name</th>
      <th>Salary</th>
    </tr>
    <tr *ngFor="let employee of employees">
      <td>{{employee.empid}}</td>
      <td>{{employee.empname}}</td>
      <td>{{employee.salary}}</td>
    </tr>
  </table>
</div>
```

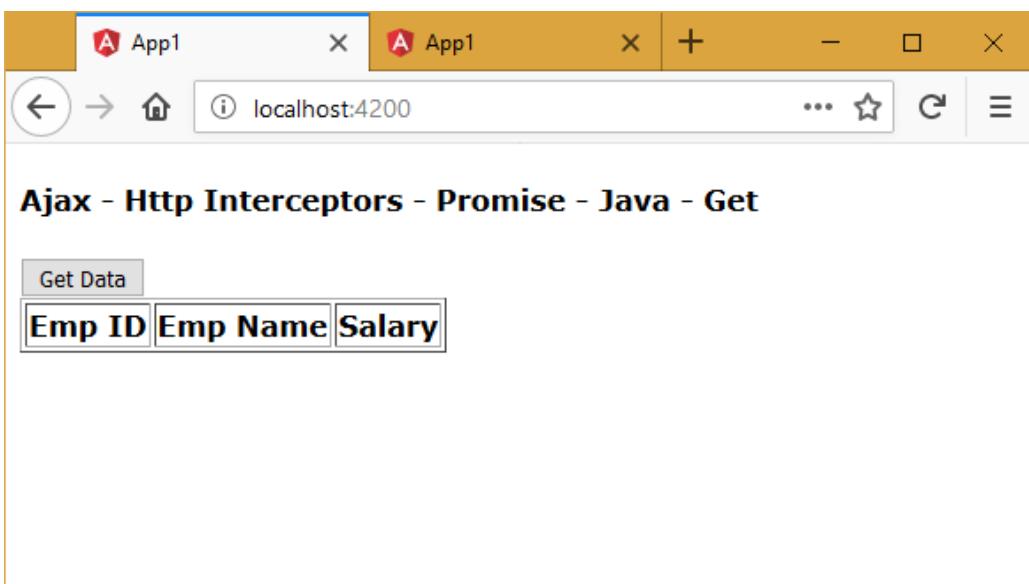
Executing the application:

- Open Command Prompt and enter the following commands:

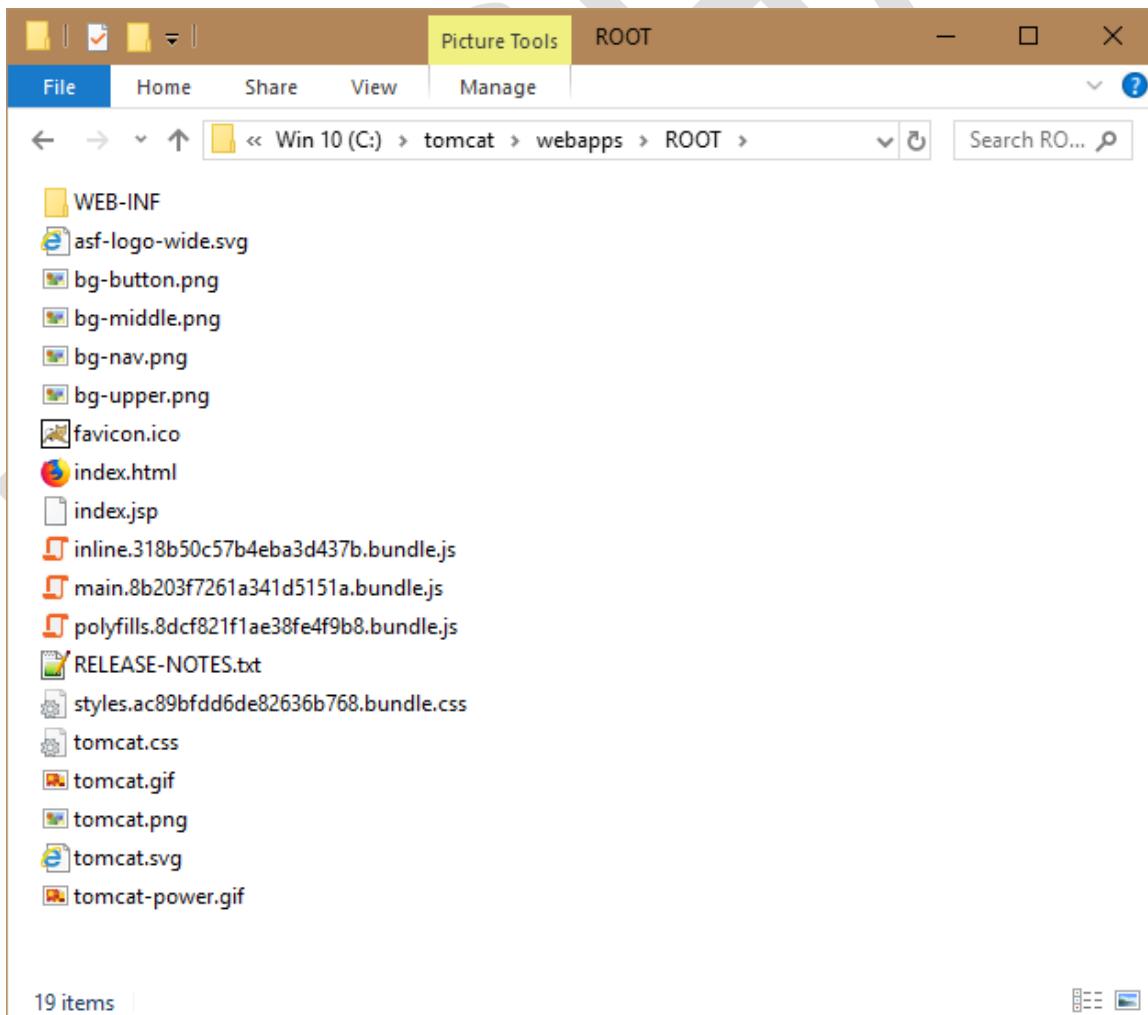
```
cd c:\tomcat\webapps\ROOT\WEB-INF\classes  
javac -cp c:\tomcat\lib\servlet-api.jar SampleServlet.java  
cd c:\angular\app1  
ng build --prod  
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



- Copy all files from "c:\angular\app1\dist" folder to "c:\tomcat\webapps\ROOT".



- Open the browser and enter the following URL:
<http://localhost:8080/index.html>

Ajax - Http Interceptors - Promise - Java - Get

| Emp ID | Emp Name | Salary |
|--------|----------|--------|
| 1 | Scott | 4000 |
| 2 | Allen | 7500 |
| 3 | Jones | 9200 |
| 4 | James | 8400 |
| 5 | Smith | 5600 |

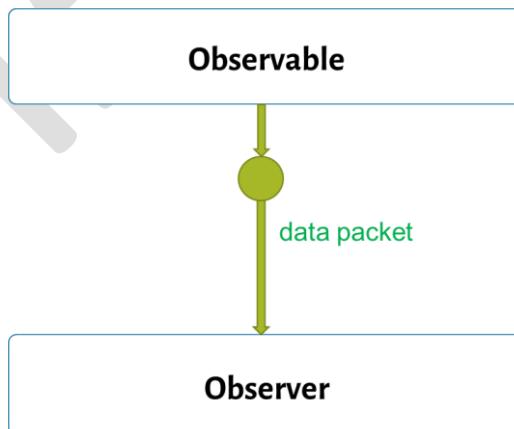
Get Data

Click on “Get Data”.

RxJS

Observable and Observer

- Observable is a data source (User events, Http requests, Custom data source), which emits data packets to the observer.
- Observer is an object that subscribes to the observable and listens to the data packets that are sent by observables.

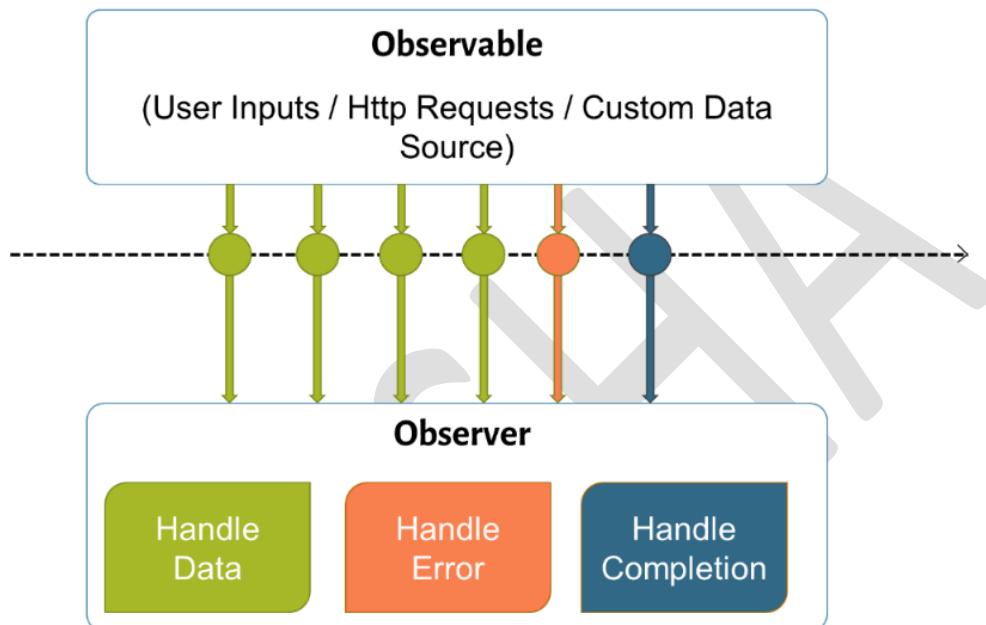


Data Flow

- “Observable” emits (sends) data to observer. Everytime it emits data, the corresponding callback function automatically executes in observer.

Types of Callback functions in Observer:

- **HandleData Callback:** It executes when next data packet has been emitted by the observable.
- **HandleError Callback:** It executes when error emitted by the observable. If this is invoked, the observable stream ends; that means, no other callback functions are invoked further.
- **HandleCompletion Callback:** It executes when observable indicates stream completion. If this is invoked, the observable stream ends; that means, no other callback functions are invoked further.

**Creating Observable****Import “Observable”:**

```
import { Observable } from "rxjs/Observable";
```

Import RxJS Operators:

```
import "rxjs/rx";
```

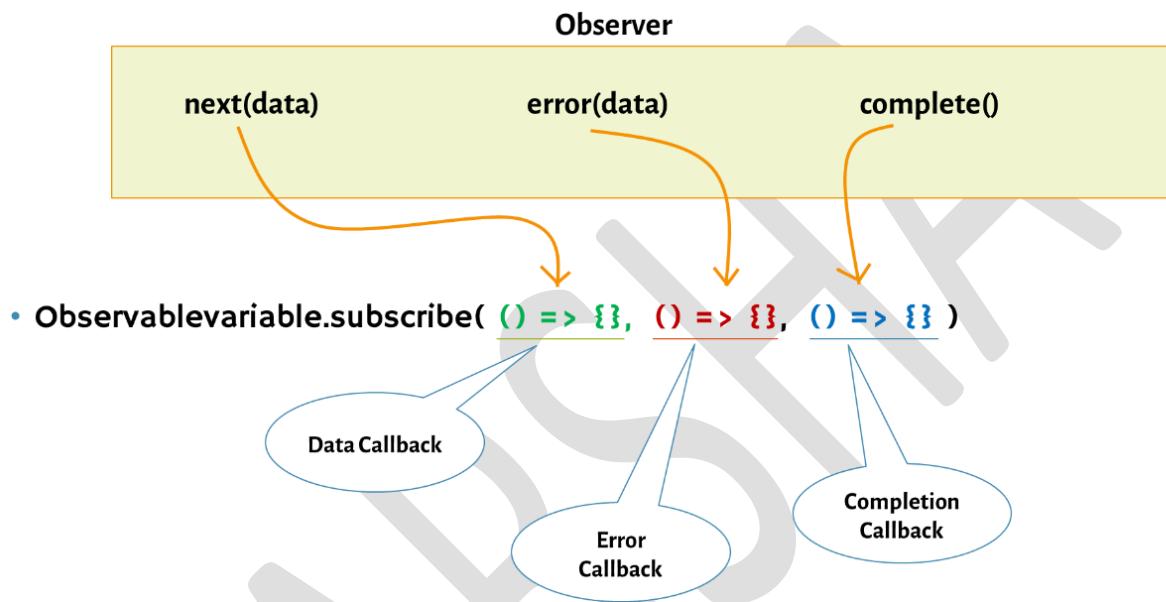
Creating Observable:

1. Observable.interval()
2. Observable.range()
3. Observable.from()
4. Observable.fromEvent()
5. Observable.create()
6. Subject

Creating Observer

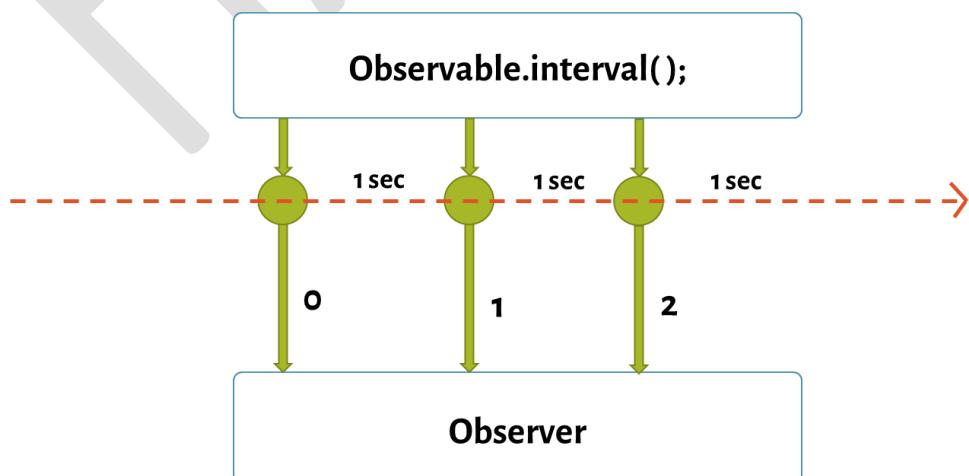
```
Observablevariable.subscribe( ()=>{ ... }, ()=>{ ... }, ()=>{ ... })
```

Note: The three callback functions are respectively “data callback”, “error callback” and “completion callback”. The data callback executes when the observable emits (passes) some data to the observer. The error callback executes when the observable emits (passes) some error; but all further callbacks will be stopped in this case. The completion callback executes when the observable emits (passes) completion to observer; so further callbacks will be stopped in this case also.



Observable.interval

- It emits (passes) a number (0 to unlimited) to the observer, for every completion of specified no. of milli seconds.
- It is the easy way to create observable. For every specified no. of milli seconds, it invokes the observer. The observer receives a new value.



Creating Observable.interval

Observable.interval(milli seconds);

Observable.Interval - Example

Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular  
ng new app1  
cd c:\angular\app1  
ng g component Home  
ng g component About  
ng g component Contact
```

c:\angular\app1\package.json

```
{  
  "name": "app1",  
  "version": "0.0.0",  
  "license": "MIT",  
  "scripts": {  
    "ng": "ng",  
    "start": "ng serve",  
    "build": "ng build --prod",  
    "test": "ng test",  
    "lint": "ng lint",  
    "e2e": "ng e2e"  
  },  
  "private": true,  
  "dependencies": {  
    "@angular/animations": "^5.2.0",  
    "@angular/common": "^5.2.0",  
    "@angular/compiler": "^5.2.0",  
    "@angular/core": "^5.2.0",  
    "@angular/forms": "^5.2.0",  
    "@angular/http": "^5.2.0",  
    "@angular/platform-browser": "^5.2.0",  
    "@angular/platform-browser-dynamic": "^5.2.0",  
    "@angular/router": "^5.2.0",  
    "core-js": "^2.4.1",  
    "rxjs": "^5.5.6",  
    "zone.js": "^0.8.19"  
  },  
  "devDependencies": {  
    "@angular/cli": "~1.7.4",  
    "@angular/compiler-cli": "^5.2.0",  
    "@angular/language-service": "^5.2.0",  
    "@types/jasmine": "~2.8.3",  
    "codelyzer": "3.2.1",  
    "jasmine-core": "3.1.0",  
    "jasmine-spec-reporter": "4.1.0",  
    "karma": "3.0.0",  
    "karma-chrome-launcher": "2.2.2",  
    "karma-jasmine": "2.0.1",  
    "karma-jasmine-html-reporter": "1.1.0",  
    "protractor": "5.4.2",  
    "ts-node": "7.0.1",  
    "tslint": "5.11.0",  
    "typescript": "3.1.6"  
  }  
}
```

```
"@types/jasminewd2": "~2.0.2",
"@types/node": "~6.0.60",
"codelyzer": "^4.0.1",
"jasmine-core": "~2.8.0",
"jasmine-spec-reporter": "~4.2.1",
"karma": "~2.0.0",
"karma-chrome-launcher": "~2.2.0",
"karma-coverage-istanbul-reporter": "^1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\styles.css

```
#container
{
    background-color: #ccccff;
    margin: 5px;
    padding: 5px;
    border-radius: 5px;
    height: 100px;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";
import { Routes, RouterModule } from "@angular/router";
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { ContactComponent } from './contact/contact.component';

var myroutes: Routes = [
    { path: "", component: HomeComponent },
    { path: "home", component: HomeComponent },
    { path: "about", component: AboutComponent },
    { path: "contact", component: ContactComponent }
];
var myroutes2 = RouterModule.forRoot(myroutes);

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    AboutComponent,
    ContactComponent
]
```

```
],
imports: [
  BrowserModule, FormsModule, myroutes2
],
providers: [],
bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";
import { FormGroup, FormControl, Validators } from "@angular/forms";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div class="class1">
  <h4>Routing</h4>
  <a routerLink="home">Home</a>
  <a routerLink="about">About</a>
  <a routerLink="contact">Contact</a>
  <div id="container">
    <router-outlet>
    </router-outlet>
  </div>
</div>
```

c:\angular\app1\src\app\home\home.component.ts

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Observable } from "rxjs/Observable";
import "rxjs/Rx";
import { Subscription } from "rxjs/Subscription";

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit, OnDestroy
{
  subscription: Subscription;

  ngOnInit()
  {
```

```
var mynumbers = Observable.interval(1000);
this.subscription = mynumbers.subscribe(
  (n) =>
  {
    console.log(n);
  },
  (error) =>
  {
    console.log(error);
  },
  () =>
  {
    console.log("Completed");
  });
}

ngOnDestroy()
{
  this.subscription.unsubscribe();
}
}
```

c:\angular\app1\src\app\home\home.component.html

```
<div class="class1">
<h5>Home</h5>
</div>
```

c:\angular\app1\src\app\about\about.component.ts

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

@Component({
  selector: 'app-about',
  templateUrl: './about.component.html',
  styleUrls: ['./about.component.css']
})
export class AboutComponent implements OnInit
{

  constructor() { }

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\about\about.component.html

```
<div class="class1">
<h5>About</h5>
</div>
```

c:\angular\app1\src\app\contact\contact.component.ts

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

@Component({
  selector: 'app-contact',
  templateUrl: './contact.component.html',
  styleUrls: ['./contact.component.css']
})
export class ContactComponent implements OnInit {

  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\contact\contact.component.html

```
<div class="class1">
  <h5>Contact</h5>
</div>
```

Executing the application:

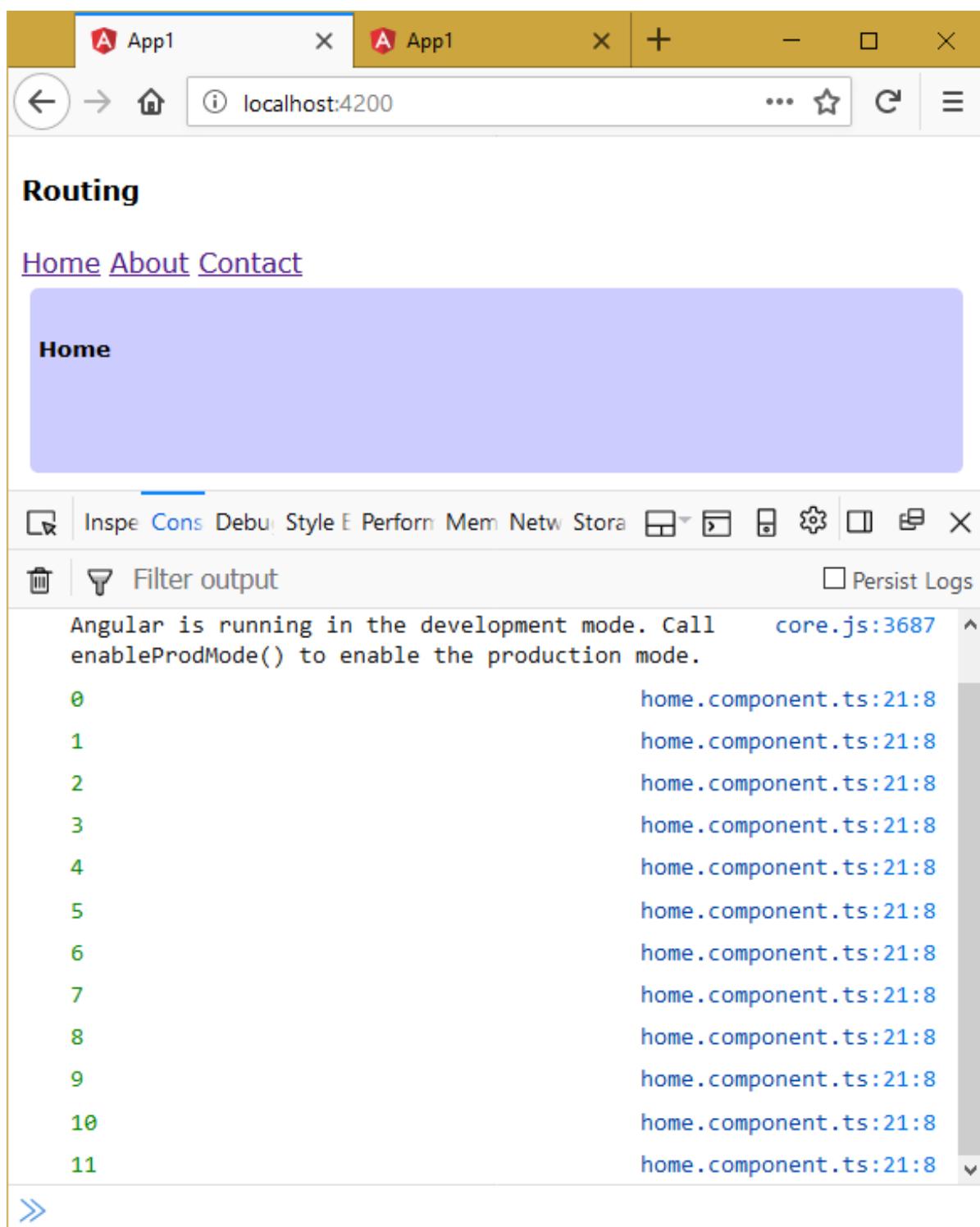
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

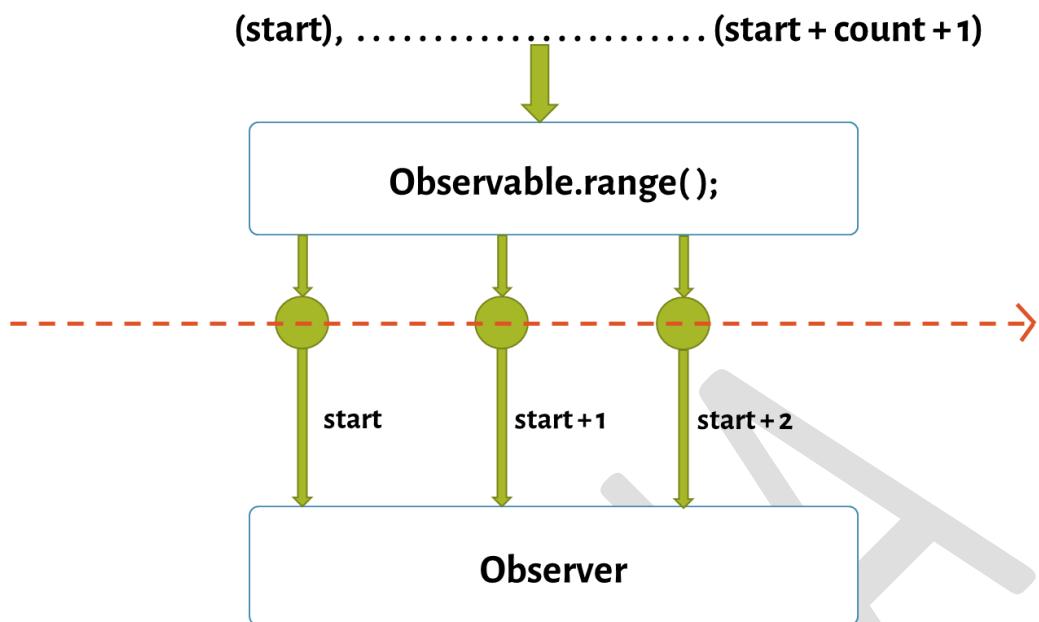
- Open the browser and enter the following URL:

```
http://localhost:4200
```



Observable.range

- It emits (passes) a number (beginning with start number, incremented by one, upto the specified count of numbers) to the observer.



Creating Observable.range

```
Observable.range(start, count);
```

Observable.range - Example

Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g component Home
ng g component About
ng g component Contact
```

c:\angular\app1\package.json

```
{
  "name": "app1",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build --prod",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
}
```

```
"private": true,  
"dependencies": {  
  "@angular/animations": "^5.2.0",  
  "@angular/common": "^5.2.0",  
  "@angular/compiler": "^5.2.0",  
  "@angular/core": "^5.2.0",  
  "@angular/forms": "^5.2.0",  
  "@angular/http": "^5.2.0",  
  "@angular/platform-browser": "^5.2.0",  
  "@angular/platform-browser-dynamic": "^5.2.0",  
  "@angular/router": "^5.2.0",  
  "core-js": "^2.4.1",  
  "rxjs": "^5.5.6",  
  "zone.js": "^0.8.19"  
},  
"devDependencies": {  
  "@angular/cli": "~1.7.4",  
  "@angular/compiler-cli": "^5.2.0",  
  "@angular/language-service": "^5.2.0",  

```

c:\angular\app1\src\styles.css

```
#container  
{  
  background-color: #ccccff;  
  margin: 5px;  
  padding: 5px;  
  border-radius: 5px;  
  height: 100px;  
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';  
import { NgModule } from '@angular/core';  
import { AppComponent } from './app.component';
```

```

import { FormsModule } from "@angular/forms";
import { Routes, RouterModule } from "@angular/router";
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { ContactComponent } from './contact/contact.component';

var myroutes: Routes = [
  { path: "", component: HomeComponent },
  { path: "home", component: HomeComponent },
  { path: "about", component: AboutComponent },
  { path: "contact", component: ContactComponent }
];
var myroutes2 = RouterModule.forRoot(myroutes);

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    AboutComponent,
    ContactComponent
  ],
  imports: [
    BrowserModule, FormsModule, myroutes2
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```

import { Component } from "@angular/core";
import { FormGroup, FormControl, Validators } from "@angular/forms";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```

<div class="class1">
  <h4>Routing</h4>
  <a routerLink="home">Home</a>
  <a routerLink="about">About</a>
  <a routerLink="contact">Contact</a>
  <div id="container">
    <router-outlet>
    </router-outlet>
```

```
</div>
</div>
```

c:\angular\app1\src\app\home\home.component.ts

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Observable } from "rxjs/Observable";
import "rxjs/Rx";
import { Subscription } from "rxjs/Subscription";

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit, OnDestroy
{
  subscription: Subscription;

  ngOnInit()
  {
    var mynumbers = Observable.range(11, 20);
    this.subscription = mynumbers.subscribe(
      (n) =>
      {
        console.log(n);
      },
      (error) =>
      {
        console.log(error);
      },
      () =>
      {
        console.log("Completed");
      });
  }

  ngOnDestroy()
  {
    this.subscription.unsubscribe();
  }
}
```

c:\angular\app1\src\app\home\home.component.html

```
<div class="class1">
  <h5>Home</h5>
</div>
```

c:\angular\app1\src\app\about\about.component.ts

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

@Component({
```

```
    selector: 'app-about',
    templateUrl: './about.component.html',
    styleUrls: ['./about.component.css']
)
export class AboutComponent implements OnInit
{
  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\about\about.component.html

```
<div class="class1">
  <h5>About</h5>
</div>
```

c:\angular\app1\src\app\contact\contact.component.ts

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

@Component({
  selector: 'app-contact',
  templateUrl: './contact.component.html',
  styleUrls: ['./contact.component.css']
})
export class ContactComponent implements OnInit
{
  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\contact\contact.component.html

```
<div class="class1">
  <h5>Contact</h5>
</div>
```

Executing the application:

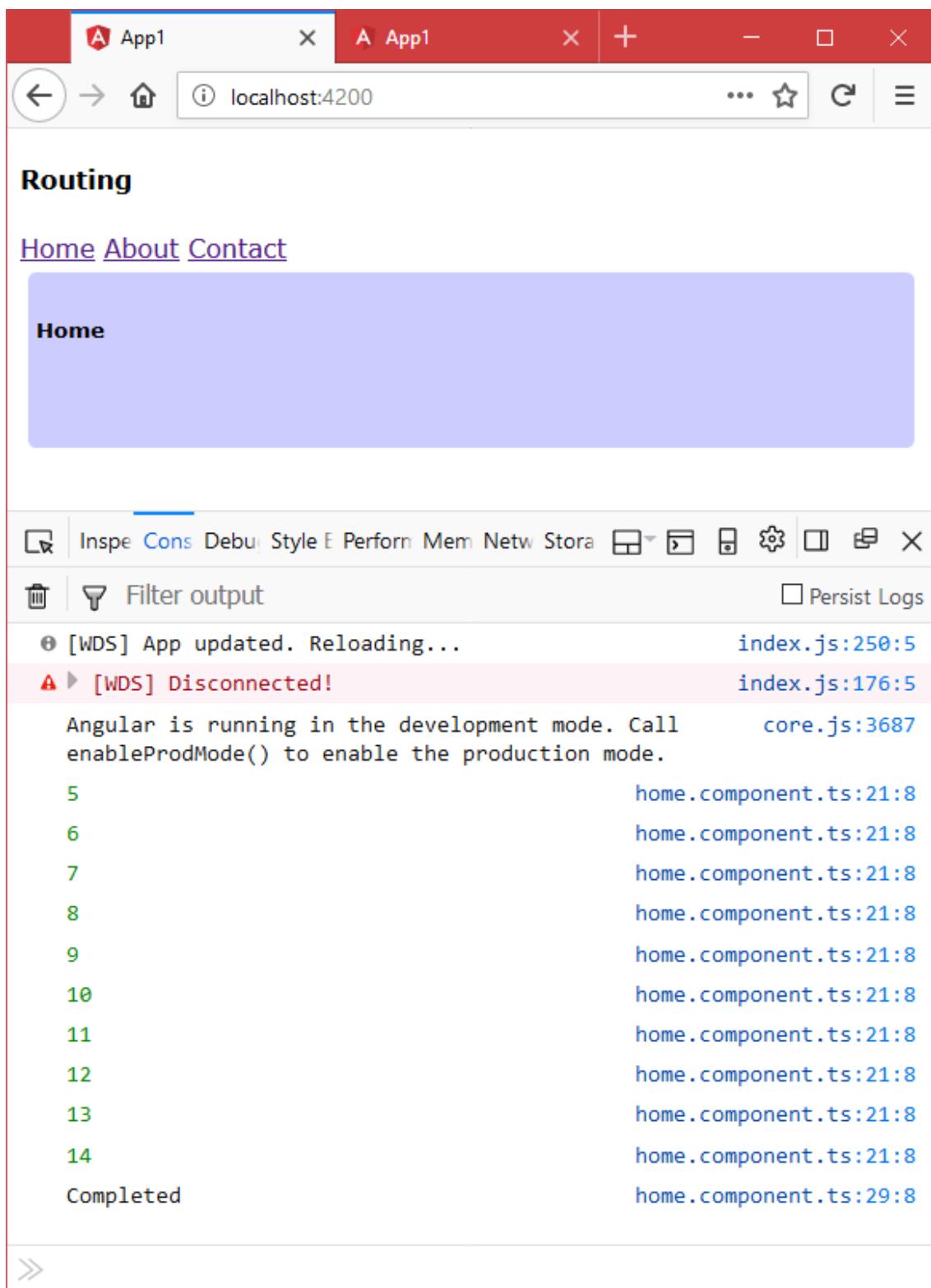
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

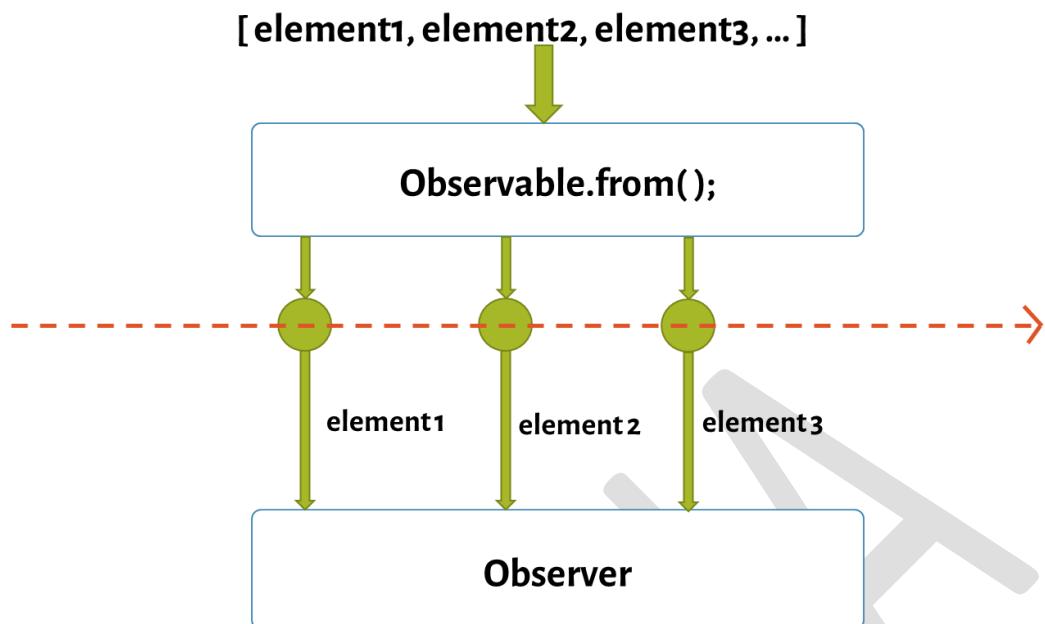
- Open the browser and enter the following URL:

```
http://localhost:4200
```



Observable.from

- It emits (passes) each element of an array one-by-one, from first to last element, to the observer.



Creating Observable.from

`Observable.from(array);`

Observable.from - Example

Creating Application

- Open Command Prompt and enter the following commands:

```

cd c:\angular
ng new app1
cd c:\angular\app1
ng g component Home
ng g component About
ng g component Contact
  
```

c:\angular\app1\package.json

```

{
  "name": "app1",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build --prod",
    "test": "ng test",
    "lint": "ng lint",
  }
}
  
```

```
"e2e": "ng e2e"
},
"private": true,
"dependencies": {
  "@angular/animations": "^5.2.0",
  "@angular/common": "^5.2.0",
  "@angular/compiler": "^5.2.0",
  "@angular/core": "^5.2.0",
  "@angular/forms": "^5.2.0",
  "@angular/http": "^5.2.0",
  "@angular/platform-browser": "^5.2.0",
  "@angular/platform-browser-dynamic": "^5.2.0",
  "@angular/router": "^5.2.0",
  "core-js": "^2.4.1",
  "rxjs": "^5.5.6",
  "zone.js": "^0.8.19"
},
"devDependencies": {
  "@angular/cli": "~1.7.4",
  "@angular/compiler-cli": "^5.2.0",
  "@angular/language-service": "^5.2.0",
  "@types/jasmine": "~2.8.3",
  "@types/jasminewd2": "~2.0.2",
  "@types/node": "~6.0.60",
  "codelyzer": "^4.0.1",
  "jasmine-core": "~2.8.0",
  "jasmine-spec-reporter": "~4.2.1",
  "karma": "~2.0.0",
  "karma-chrome-launcher": "~2.2.0",
  "karma-coverage-istanbul-reporter": "^1.2.1",
  "karma-jasmine": "~1.1.0",
  "karma-jasmine-html-reporter": "^0.2.2",
  "protractor": "~5.1.2",
  "ts-node": "~4.1.0",
  "tslint": "~5.9.1",
  "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\styles.css

```
#container
{
  background-color: #ccccff;
  margin: 5px;
  padding: 5px;
  border-radius: 5px;
  height: 100px;
}
```

c:\angular\app1\src\app\app.module.ts

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

```

import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { ContactComponent } from './contact/contact.component';

var myroutes: Routes = [
  { path: "", component: HomeComponent },
  { path: "home", component: HomeComponent },
  { path: "about", component: AboutComponent },
  { path: "contact", component: ContactComponent }
];
var myroutes2 = RouterModule.forRoot(myroutes);

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    AboutComponent,
    ContactComponent
  ],
  imports: [
    BrowserModule, FormsModule, myroutes2
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}

```

c:\angular\app1\src\app\app.component.ts

```

import { Component } from "@angular/core";
import { FormGroup, FormControl, Validators } from "@angular/forms";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{
}

```

c:\angular\app1\src\app\app.component.html

```

<div class="class1">
  <h4>Routing</h4>
  <a routerLink="home">Home</a>
  <a routerLink="about">About</a>
  <a routerLink="contact">Contact</a>
  <div id="container">

```

```
<router-outlet>
</router-outlet>
</div>
</div>
```

c:\angular\app1\src\app\home\home.component.ts

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Observable } from "rxjs/Observable";
import "rxjs/Rx";
import { Subscription } from "rxjs/Subscription";

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit, OnDestroy
{
  subscription: Subscription;

  ngOnInit()
  {
    var myarray = [10, 20, 30, 40, 50];
    var mynumbers = Observable.from(myarray);
    this.subscription = mynumbers.subscribe(
      (n) =>
      {
        console.log(n);
      },
      (error) =>
      {
        console.log(error);
      },
      () =>
      {
        console.log("Completed");
      });
  }

  ngOnDestroy()
  {
    this.subscription.unsubscribe();
  }
}
```

c:\angular\app1\src\app\home\home.component.html

```
<div class="class1">
<h5>Home</h5>
</div>
```

c:\angular\app1\src\app\about\about.component.ts

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

@Component({
  selector: 'app-about',
  templateUrl: './about.component.html',
  styleUrls: ['./about.component.css']
})
export class AboutComponent implements OnInit
{
  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\about\about.component.html

```
<div class="class1">
  <h5>About</h5>
</div>
```

c:\angular\app1\src\app\contact\contact.component.ts

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

@Component({
  selector: 'app-contact',
  templateUrl: './contact.component.html',
  styleUrls: ['./contact.component.css']
})
export class ContactComponent implements OnInit
{
  constructor() {}

  ngOnInit()
  {
  }
}
```

c:\angular\app1\src\app\contact\contact.component.html

```
<div class="class1">
  <h5>Contact</h5>
</div>
```

Executing the application:

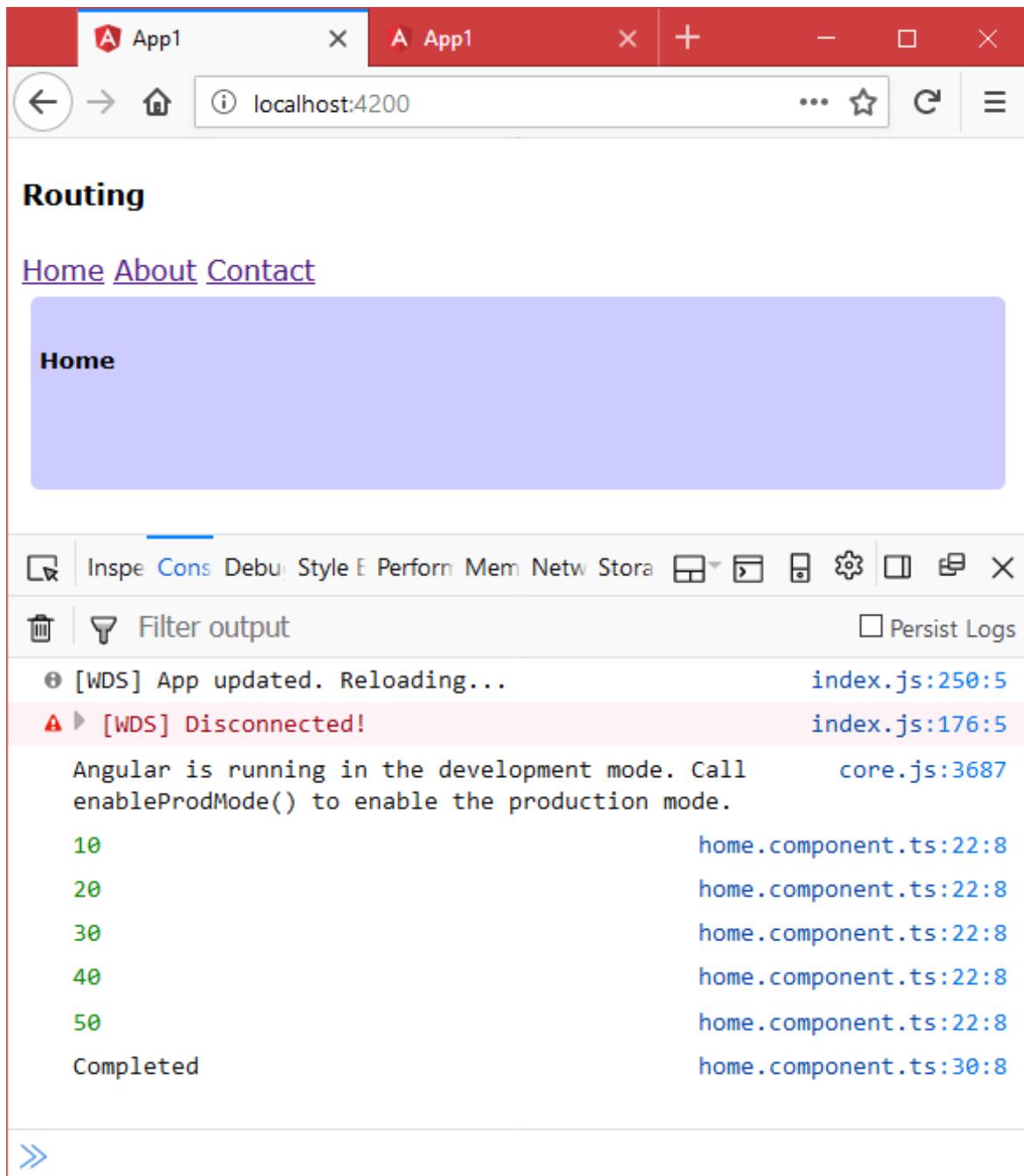
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

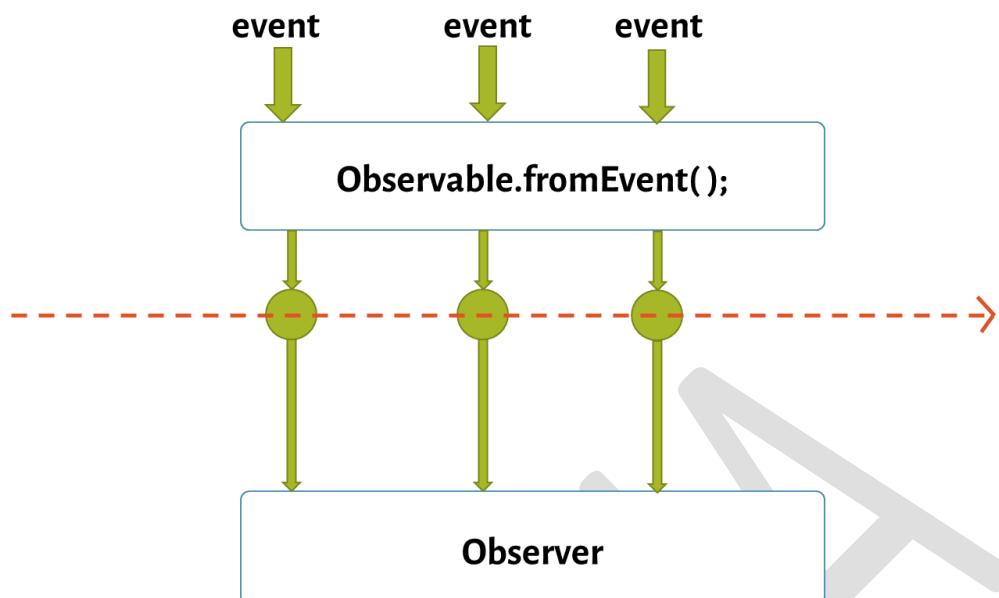
- Open the browser and enter the following URL:

http://localhost:4200



Observable.fromEvent

- It emits (passes) “event arguments” to the observer, everytime the event is raised.



Creating Observable.fromEvent

```
Observable.fromEvent(document.getElementById("element"), "event");
```

Observable.fromEvent - Example

Creating Application

- Open Command Prompt and enter the following commands:
- ```

cd c:\angular
ng new app1
cd c:\angular\app1
ng g component Home
ng g component About
ng g component Contact

```

#### c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 }
}
```

```
},
"private": true,
"dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
},
"devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\styles.css

```
#container
{
 background-color: #ccccff;
 margin: 5px;
 padding: 5px;
 border-radius: 5px;
 height: 100px;
}
```

c:\angular\app1\src\app\app.module.ts

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

```

import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";
import { Routes, RouterModule } from "@angular/router";
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { ContactComponent } from './contact/contact.component';

var myroutes: Routes = [
 { path: "", component: HomeComponent },
 { path: "home", component: HomeComponent },
 { path: "about", component: AboutComponent },
 { path: "contact", component: ContactComponent }
];
var myroutes2 = RouterModule.forRoot(myroutes);

@NgModule({
 declarations: [
 AppComponent,
 HomeComponent,
 AboutComponent,
 ContactComponent
],
 imports: [
 BrowserModule, FormsModule, myroutes2
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```

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

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```

<div class="class1">
 <h4>Routing</h4>
 Home
 About
 Contact
 <div id="container">
 <router-outlet>
 </router-outlet>
```

```
</div>
</div>
```

c:\angular\app1\src\app\home\home.component.ts

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Observable } from "rxjs/Observable";
import "rxjs/Rx";
import { Subscription } from "rxjs/Subscription";

@Component({
 selector: 'app-home',
 templateUrl: './home.component.html',
 styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit, OnDestroy
{
 subscription: Subscription;

 ngOnInit()
 {
 var txtClick = Observable.fromEvent(document.getElementById("txt1"), "keyup");
 this.subscription = txtClick.subscribe(
 (n) =>
 {
 console.log(n);
 },
 (error) =>
 {
 console.log(error);
 },
 () =>
 {
 console.log("Completed");
 });
 }

 ngOnDestroy()
 {
 this.subscription.unsubscribe();
 }
}
```

c:\angular\app1\src\app\home\home.component.html

```
<div class="class1">
 <h5>Home</h5>
 <input type="text" id="txt1">
</div>
```

c:\angular\app1\src\app\about\about.component.ts

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

```
@Component({
 selector: 'app-about',
 templateUrl: './about.component.html',
 styleUrls: ['./about.component.css']
})
export class AboutComponent implements OnInit
{
 constructor() {}

 ngOnInit()
 {
 }
}
```

c:\angular\app1\src\app\about\about.component.html

```
<div class="class1">
 <h5>About</h5>
</div>
```

c:\angular\app1\src\app\contact\contact.component.ts

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

@Component({
 selector: 'app-contact',
 templateUrl: './contact.component.html',
 styleUrls: ['./contact.component.css']
})
export class ContactComponent implements OnInit
{
 constructor() {}

 ngOnInit()
 {
 }
}
```

c:\angular\app1\src\app\contact\contact.component.html

```
<div class="class1">
 <h5>Contact</h5>
</div>
```

### Executing the application:

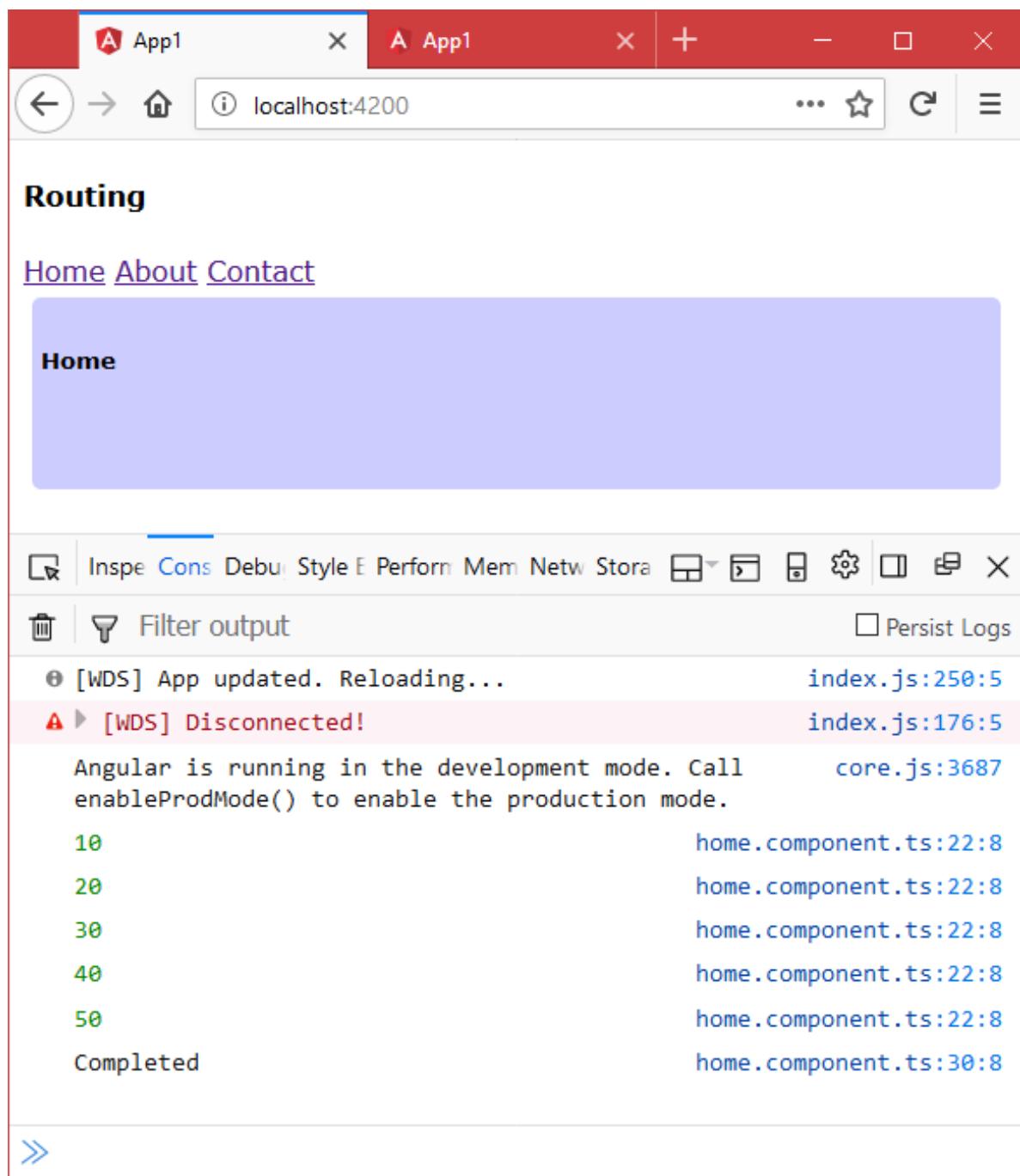
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

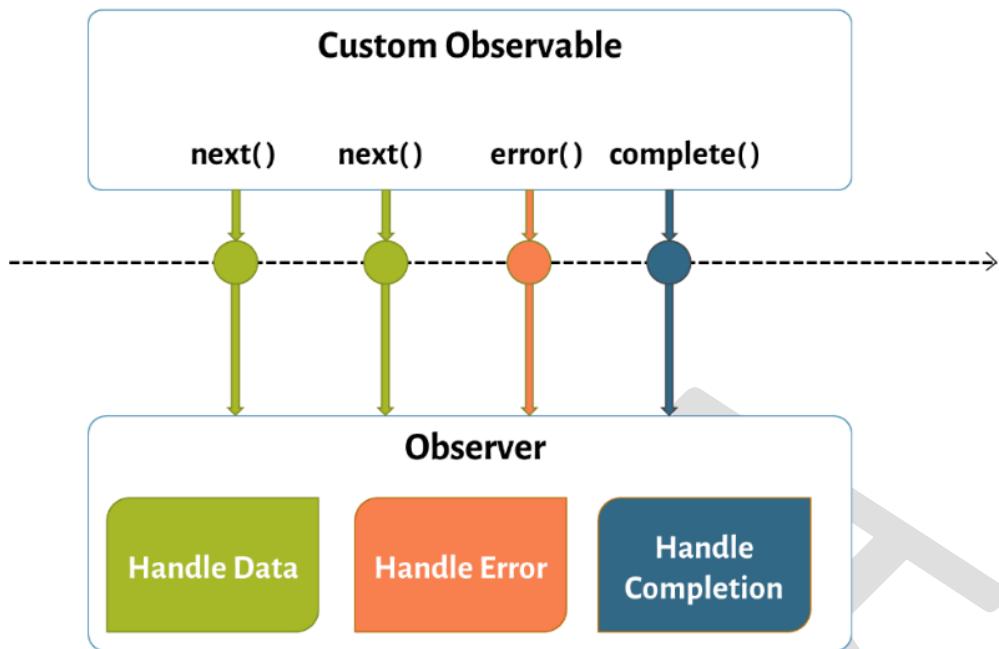
- Open the browser and enter the following URL:

```
http://localhost:4200
```



### Custom Observables

- It emits (passes) “event arguments” to the observer, everytime the event is raised.



### Creating Custom Observables

```
import { Observer } from "rxjs/Observer";
Observable.create(observer : Observer)=> {
 observer.next(data); //passes data to "handle data" function
 observer.error(data); //passes data to "handle error" function
 observer.complete(); //finishes the observable and calls "handle completion" function
};
```

### Custom Observables - Example

#### Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g component Home
ng g component About
ng g component Contact
```

#### c:\angular\app1\package.json

```
{
 "name": "app1",
```

```
"version": "0.0.0",
"license": "MIT",
"scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
},
"private": true,
"dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
},
"devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\styles.css

---

```
#container
{
 background-color: #ccccff;
 margin: 5px;
```

```
padding: 5px;
border-radius: 5px;
height: 100px;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";
import { Routes, RouterModule } from "@angular/router";
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { ContactComponent } from './contact/contact.component';

var myroutes: Routes = [
 { path: "", component: HomeComponent },
 { path: "home", component: HomeComponent },
 { path: "about", component: AboutComponent },
 { path: "contact", component: ContactComponent }
];
var myroutes2 = RouterModule.forRoot(myroutes);

@NgModule({
 declarations: [
 AppComponent,
 HomeComponent,
 AboutComponent,
 ContactComponent
],
 imports: [
 BrowserModule, FormsModule, myroutes2
],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{
}
```

c:\angular\app1\src\app\app.component.html

```
<div class="class1">
 <h4>Routing</h4>
 Home
 About
 Contact
 <div id="container">
 <router-outlet>
 </router-outlet>
 </div>
</div>
```

c:\angular\app1\src\app\home\home.component.ts

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Observable } from "rxjs/Observable";
import { Observer } from "rxjs/Observer";
import "rxjs/Rx";
import { Subscription } from "rxjs/Subscription";

@Component({
 selector: 'app-home',
 templateUrl: './home.component.html',
 styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit, OnDestroy
{
 subscription: Subscription;

 ngOnInit()
 {
 var myObservable = Observable.create((observer: Observer<string>) =>
 {
 setTimeout(() => { observer.next("first data package"); }, 2000);
 setTimeout(() => { observer.next("second data package"); }, 5000);
 setTimeout(() => { observer.next("third data package"); }, 8000);
 //setTimeout(() => { observer.error("error message"); }, 10000);
 setTimeout(() => { observer.complete(); }, 10000);
 setTimeout(() => { observer.next("fourth data package"); }, 12000);
 });

 this.subscription = myObservable.subscribe(
 (n) =>
 {
 console.log(n);
 },
 (error) =>
 {
 console.log(error);
 },
 () =>
 {

```

```
 console.log("Completed");
 });
}

ngOnDestroy()
{
 this.subscription.unsubscribe();
}
}
```

c:\angular\app1\src\app\home\home.component.html

```
<div class="class1">
<h5>Home</h5>
</div>
```

c:\angular\app1\src\app\about\about.component.ts

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

@Component({
 selector: 'app-about',
 templateUrl: './about.component.html',
 styleUrls: ['./about.component.css']
})
export class AboutComponent implements OnInit
{
 constructor() {}

 ngOnInit()
 {
 }
}
```

c:\angular\app1\src\app\about\about.component.html

```
<div class="class1">
<h5>About</h5>
</div>
```

c:\angular\app1\src\app\contact\contact.component.ts

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

@Component({
 selector: 'app-contact',
 templateUrl: './contact.component.html',
 styleUrls: ['./contact.component.css']
})
export class ContactComponent implements OnInit
{
 constructor() {}

 ngOnInit()
 {
 }
}
```

```
}
```

c:\angular\app1\src\app\contact\contact.component.html

```
<div class="class1">
 <h5>Contact</h5>
</div>
```

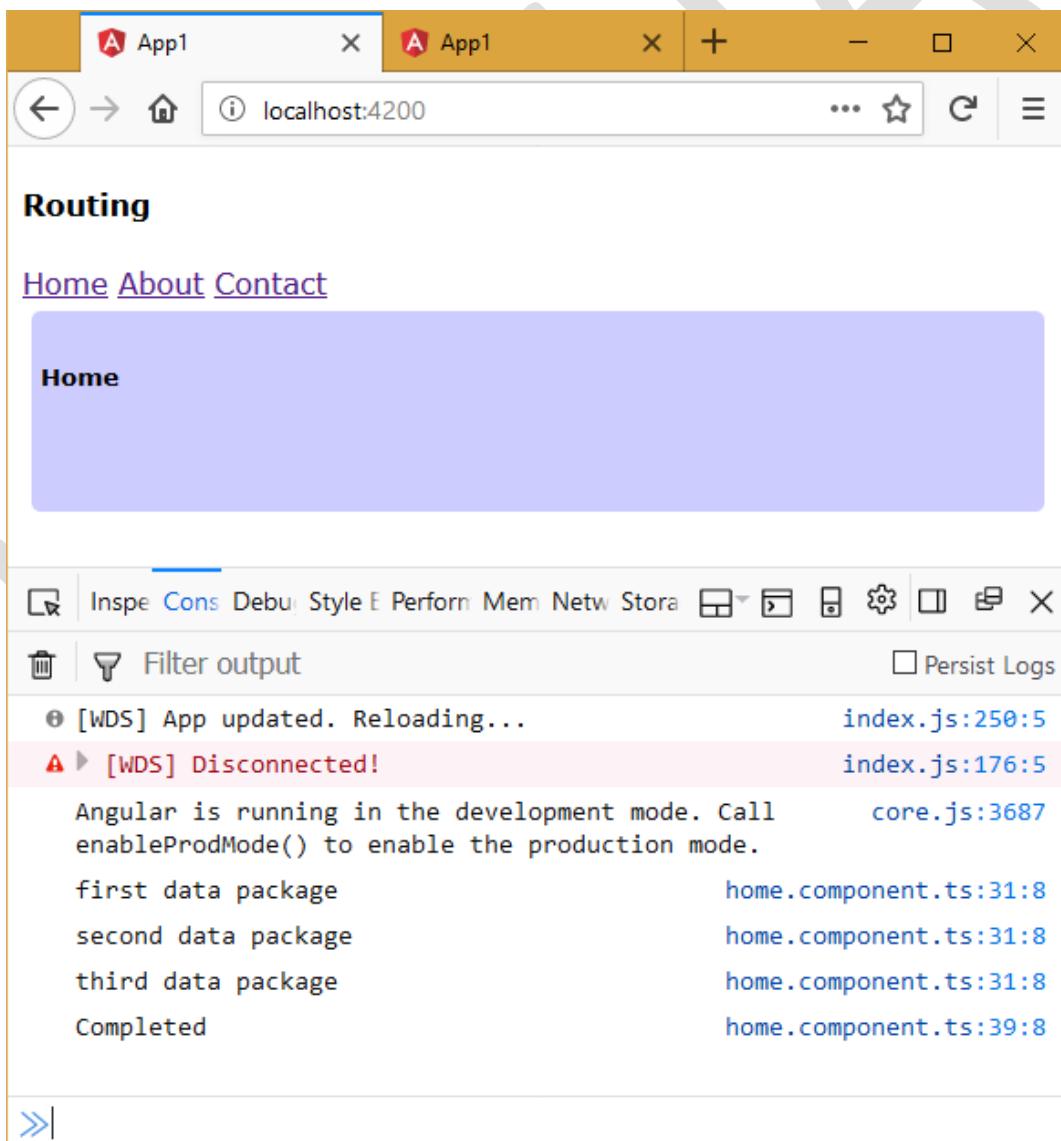
### Executing the application:

- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



## Custom Observables using Services - Example

### Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g component India
ng g component Usa
ng g service Population
```

### c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
 },
 "devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1"
 }
}
```

```
"jasmine-core": "~2.8.0",
"jasmine-spec-reporter": "~4.2.1",
"karma": "~2.0.0",
"karma-chrome-launcher": "~2.2.0",
"karma-coverage-istanbul-reporter": "^1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\population.service.ts

```
import { Injectable } from '@angular/core';
import { Observable } from "rxjs/Observable";
import { Observer } from "rxjs/Observer";

@Injectable()
export class PopulationService
{
 myObservable: Observable<number>;
 myObserver: Observer<number>;

 constructor()
 {
 this.myObservable = Observable.create((observer: Observer<number>) =>
 {
 this.myObserver = observer;
 });
 }

 next(data)
 {
 this.myObserver.next(data);
 }

 error()
 {
 this.myObserver.error("some error");
 }

 complete()
 {
 this.myObserver.complete();
 }
}
```

c:\angular\app1\src\styles.css

```
.class1
```

```
{
 background-color: #91e3e9;
 width: 500px;
 height: 150px;
 margin: 20px;
 float: left;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";
import { IndiaComponent } from './india/india.component';
import { UsaComponent } from './usa/usa.component';
import { PopulationService } from "./population.service";

@NgModule({
 declarations: [
 AppComponent,
 IndiaComponent,
 UsaComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [PopulationService],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{
}
```

c:\angular\app1\src\app\app.component.html

```
<div>
 <h4>RxJS - Custom Observables with Services</h4>
 <app-india></app-india>
 <app-usa></app-usa>
</div>
```

c:\angular\app1\src\app\india\india.component.ts

```
import { Component, OnInit, Inject } from '@angular/core';
import { PopulationService } from './population.service';

@Component({
 selector: 'app-india',
 templateUrl: './india.component.html',
 styleUrls: ['./india.component.css']
})
export class IndiaComponent {
 population: number = 1.32;

 constructor(@Inject(PopulationService) private ps : PopulationService)
 {

 }

 onIncreaseClick()
 {
 this.population++;
 this.ps.next(this.population);
 }

 onErrorClick()
 {
 this.ps.error();
 }

 onCompleteClick()
 {
 this.ps.complete();
 }
}
```

c:\angular\app1\src\app\india\india.component.html

```
<div class="class1">
<h4>India</h4>
<input type="button" value="Increase" (click)="onIncreaseClick()">
<input type="button" value="Error" (click)="onErrorClick()">
<input type="button" value="Complete" (click)="onCompleteClick()">
</div>
```

c:\angular\app1\src\app\Usa\Usa.component.ts

```
import { Component, OnInit, OnDestroy, Inject } from '@angular/core';
import { PopulationService } from './population.service';
import { Subscription } from "rxjs/Subscription";

@Component({
 selector: 'app-usa',
 templateUrl: './usa.component.html',
 styleUrls: ['./usa.component.css']
})
```

```
)
export class UsaComponent implements OnInit, OnDestroy
{
 indiapopulation: number = 1.32;
 subscription: Subscription;

 constructor(@Inject(PopulationService) private ps : PopulationService)
 {}
 {}

 ngOnInit()
 {
 this.subscription = this.ps.myObservable.subscribe(
 (n) =>
 {
 this.indiapopulation = n;
 },
 (error) =>
 {
 console.log(error);
 },
 () =>
 {
 console.log("Completed");
 });
 }

 ngOnDestroy()
 {
 this.subscription.unsubscribe();
 }
}
```

c:\angular\app1\src\app\Usa\Usa.component.html

```
<div class="class1">
 <h4>United States</h4>
 India population: {{indiapopulation}} billion.
</div>
```

#### Executing the application:

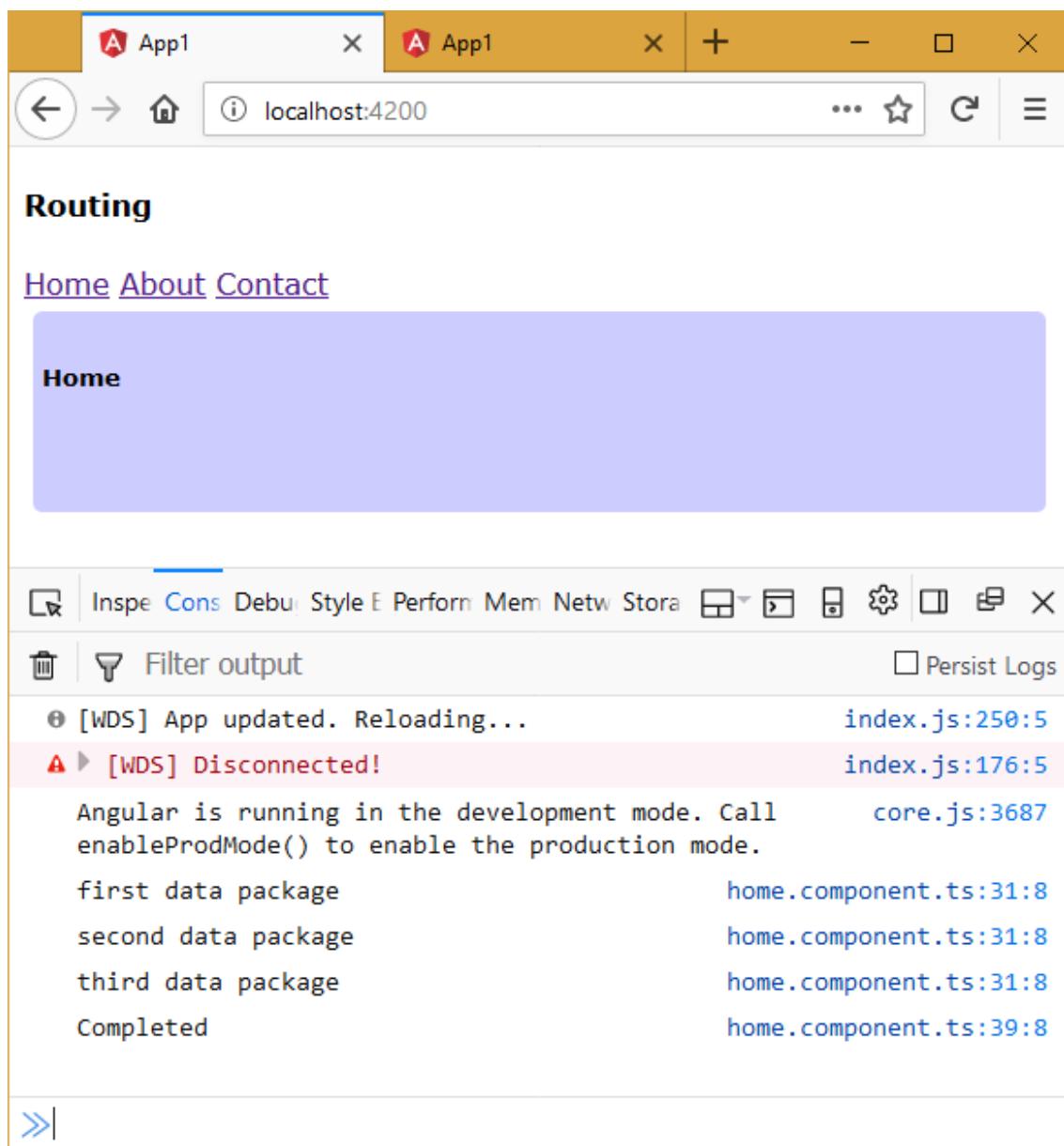
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

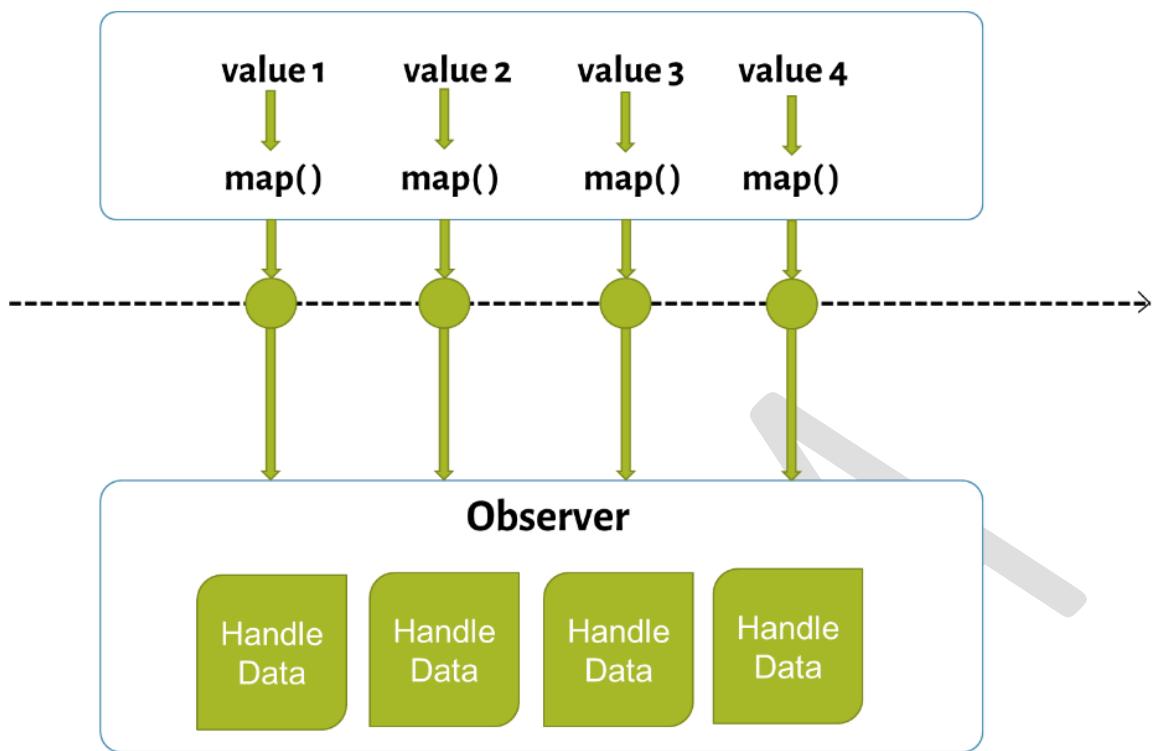
- Open the browser and enter the following URL:

```
http://localhost:4200
```



## Map

- It executes the map function every time, the observable emits a value to the observer.



### Creating Map

```
Observablevariable.map((variable) => {
 //do something with item and return it
});
```

### Map - Example

#### Creating Application

- Open Command Prompt and enter the following commands:
- ```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g component India
ng g component Usa
ng g service Population
```

c:\angular\app1\package.json

```
{
  "name": "app1",
  "version": "0.0.0",
  "license": "MIT",
```

```

"scripts": {
  "ng": "ng",
  "start": "ng serve",
  "build": "ng build --prod",
  "test": "ng test",
  "lint": "ng lint",
  "e2e": "ng e2e"
},
"private": true,
"dependencies": {
  "@angular/animations": "^5.2.0",
  "@angular/common": "^5.2.0",
  "@angular/compiler": "^5.2.0",
  "@angular/core": "^5.2.0",
  "@angular/forms": "^5.2.0",
  "@angular/http": "^5.2.0",
  "@angular/platform-browser": "^5.2.0",
  "@angular/platform-browser-dynamic": "^5.2.0",
  "@angular/router": "^5.2.0",
  "core-js": "^2.4.1",
  "rxjs": "^5.5.6",
  "zone.js": "^0.8.19"
},
"devDependencies": {
  "@angular/cli": "~1.7.4",
  "@angular/compiler-cli": "^5.2.0",
  "@angular/language-service": "^5.2.0",
  "@types/jasmine": "~2.8.3",
  "@types/jasminewd2": "~2.0.2",
  "@types/node": "~6.0.60",
  "codelyzer": "^4.0.1",
  "jasmine-core": "~2.8.0",
  "jasmine-spec-reporter": "~4.2.1",
  "karma": "~2.0.0",
  "karma-chrome-launcher": "~2.2.0",
  "karma-coverage-istanbul-reporter": "^1.2.1",
  "karma-jasmine": "~1.1.0",
  "karma-jasmine-html-reporter": "^0.2.2",
  "protractor": "~5.1.2",
  "ts-node": "~4.1.0",
  "tslint": "~5.9.1",
  "typescript": "~2.5.3"
}
}
}

```

c:\angular\app1\src\app\population.service.ts

```

import { Injectable } from '@angular/core';
import { Observable } from 'rxjs/Observable';
import { Observer } from 'rxjs/Observer';

@Injectable()
export class PopulationService

```

```
{  
  myObservable: Observable<number>;  
  myObserver: Observer<number>;  
  
  constructor()  
  {  
    this.myObservable = Observable.create((observer: Observer<number>) =>  
    {  
      this.myObserver = observer;  
    });  
  }  
  
  next(data)  
  {  
    this.myObserver.next(data);  
  }  
  
  error()  
  {  
    this.myObserver.error("some error");  
  }  
  
  complete()  
  {  
    this.myObserver.complete();  
  }  
}
```

c:\angular\app1\src\styles.css

```
.class1  
{  
  background-color:#91e3e9;  
  width: 500px;  
  height: 150px;  
  margin: 20px;  
  float:left;  
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';  
import { NgModule } from '@angular/core';  
import { AppComponent } from './app.component';  
import { FormsModule } from "@angular/forms";  
import { IndiaComponent } from './india/india.component';  
import { UsaComponent } from './usa/usa.component';  
import { PopulationService } from "./population.service";  
  
@NgModule({  
  declarations: [  
    AppComponent,  
    IndiaComponent,
```

```
    UsaComponent
  ],
  imports: [
    BrowserModule, FormsModule
  ],
  providers: [ PopulationService ],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div>
  <h4>RxJS - Custom Observables with Services with Map</h4>
  <app-india></app-india>
  <app-usa></app-usa>
</div>
```

c:\angular\app1\src\app\india\india.component.ts

```
import { Component, OnInit, Inject } from '@angular/core';
import { PopulationService } from "./population.service";

@Component({
  selector: 'app-india',
  templateUrl: './india.component.html',
  styleUrls: ['./india.component.css']
})
export class IndiaComponent
{
  population: number = 1.32;

  constructor(@Inject(PopulationService) private ps : PopulationService)
  {
  }

  onIncreaseClick()
  {
    this.population++;
    this.ps.next(this.population);
  }
}
```

```
onErrorHandler()
{
  this.ps.error();
}

onCompleteHandler()
{
  this.ps.complete();
}
}
```

c:\angular\app1\src\app\india\india.component.html

```
<div class="class1">
<h4>India</h4>
<input type="button" value="Increase" (click)="onIncreaseClick()">
<input type="button" value="Error" (click)="onErrorHandler()">
<input type="button" value="Complete" (click)="onCompleteHandler()">
</div>
```

c:\angular\app1\src\app\Usa\Usa.component.ts

```
import { Component, OnInit, OnDestroy, Inject } from '@angular/core';
import { PopulationService } from './population.service';
import { Subscription } from "rxjs/Subscription";
import "rxjs/Rx";

@Component({
  selector: 'app-usa',
  templateUrl: './usa.component.html',
  styleUrls: ['./usa.component.css']
})
export class UsaComponent implements OnInit, OnDestroy
{
  indiaPopulation: number = 1.32;
  subscription: Subscription;

  constructor(@Inject(PopulationService) private ps: PopulationService)
  {
  }

  ngOnInit()
  {
    this.subscription = this.ps.myObservable
      .map((item) =>
      {
        return item * 1000;
      })
      .subscribe(
        (n) =>
        {
          this.indiaPopulation = n;
        }
      );
  }

  ngOnDestroy()
  {
    this.subscription.unsubscribe();
  }
}
```

```
},
(error) =>
{
  console.log(error);
},
() =>
{
  console.log("Completed");
});
}

ngOnDestroy()
{
  this.subscription.unsubscribe();
}
}
```

c:\angular\app1\src\app\Usa\Usa.component.html

```
<div class="class1">
<h4>United States</h4>
India population: {{indiapopulation}} million.
</div>
```

Executing the application:

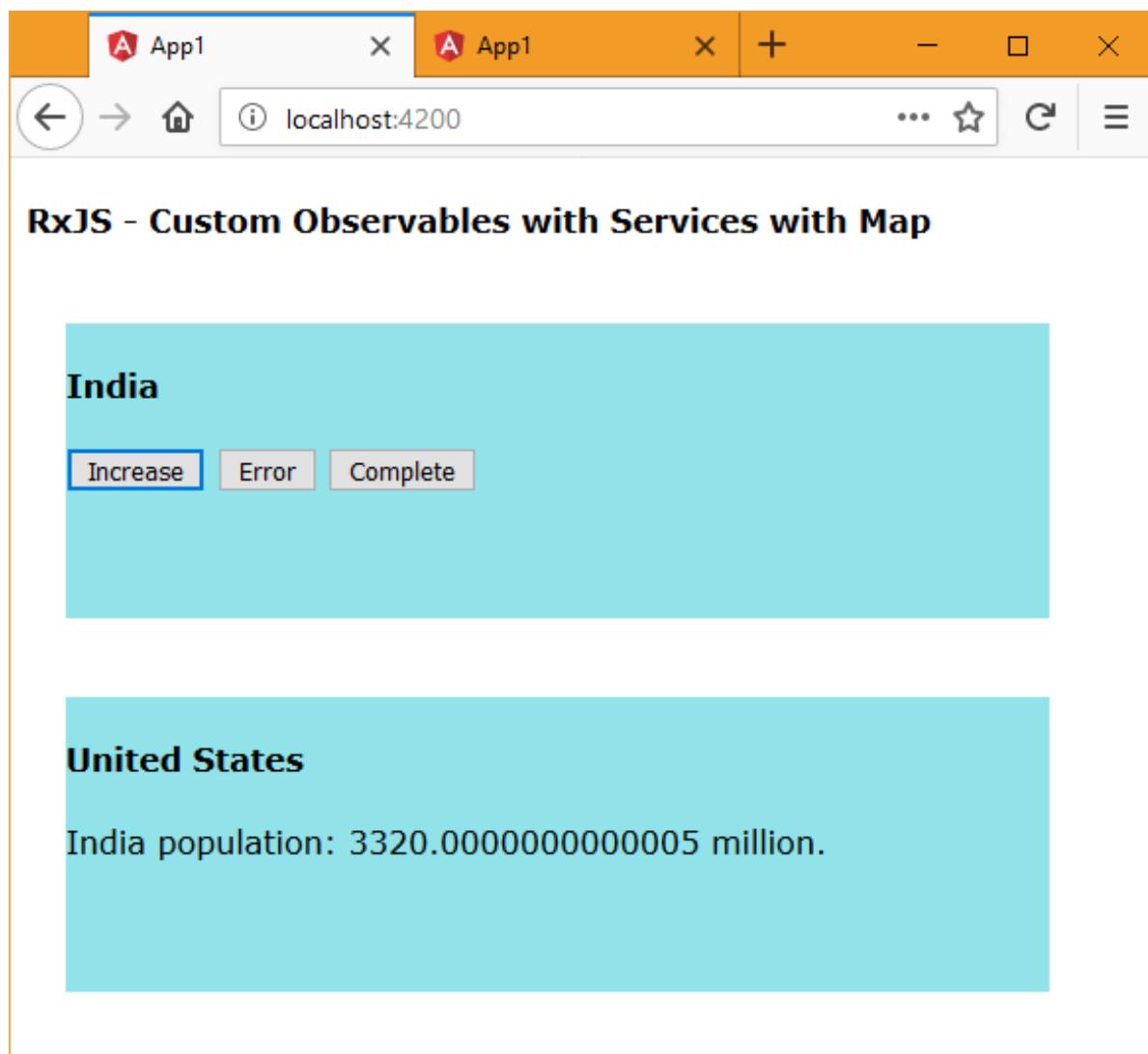
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

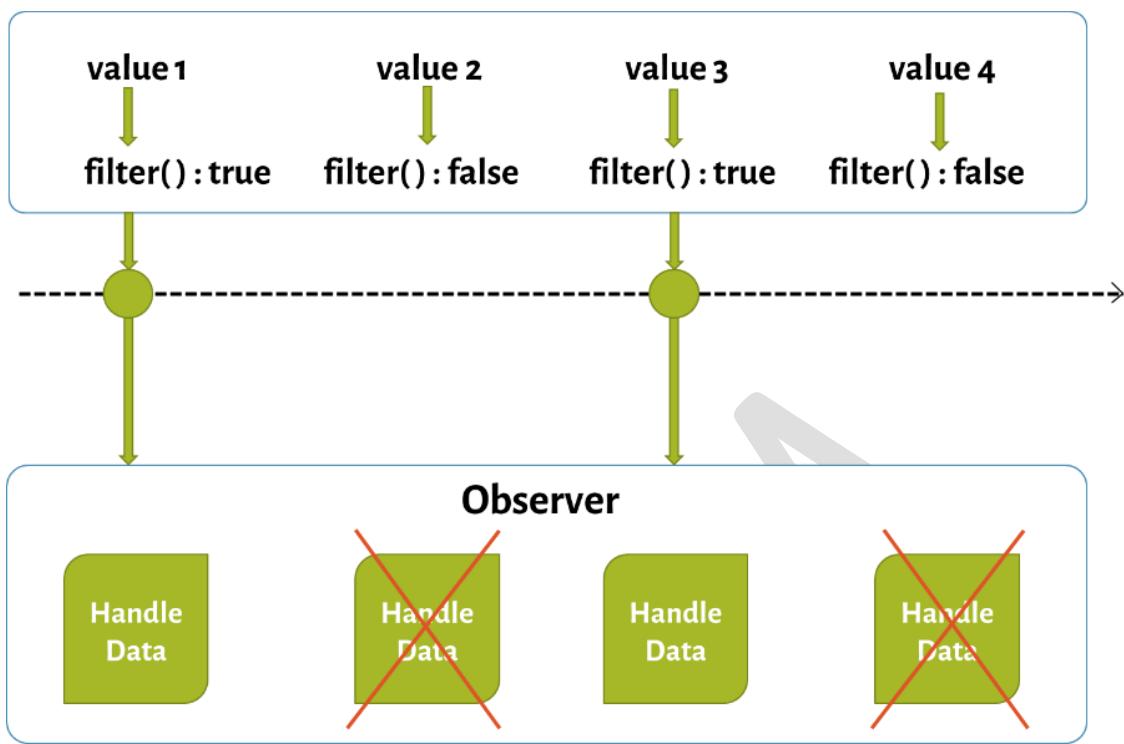
- Open the browser and enter the following URL:

```
http://localhost:4200
```



Filter

- It executes the filter function for each item when the observable emits data to the observer. If the function returns true, the value will be emitted to the observer. If the function returns false, the value will not be emitted to the observer.



Creating Filter

```
Observablevariable.filter( (variable) => {
  //do something with item and return true / false
});
```

Filter - Example

Creating Application

- Open Command Prompt and enter the following commands:
- ```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g component India
ng g component Usa
ng g service Population
```

#### c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
```

```

"scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
},
"private": true,
"dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
},
"devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
}
}
}

```

c:\angular\app1\src\app\population.service.ts

```

import { Injectable } from '@angular/core';
import { Observable } from 'rxjs/Observable';
import { Observer } from 'rxjs/Observer';

@Injectable()
export class PopulationService

```

```
{
 myObservable: Observable<number>;
 myObserver: Observer<number>;

 constructor()
 {
 this.myObservable = Observable.create((observer: Observer<number>) =>
 {
 this.myObserver = observer;
 });
 }

 next(data)
 {
 this.myObserver.next(data);
 }

 error()
 {
 this.myObserver.error("some error");
 }

 complete()
 {
 this.myObserver.complete();
 }
}
```

c:\angular\app1\src\styles.css

```
.class1
{
 background-color:#91e3e9;
 width: 500px;
 height: 150px;
 margin: 20px;
 float:left;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";
import { IndiaComponent } from './india/india.component';
import { UsaComponent } from './usa/usa.component';
import { PopulationService } from "./population.service";

@NgModule({
 declarations: [
 AppComponent,
 IndiaComponent,
```

```
 UsaComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [PopulationService],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div>
 <h4>RxJS - Custom Observables with Services with Filter</h4>
 <app-india></app-india>
 <app-usa></app-usa>
</div>
```

c:\angular\app1\src\app\india\india.component.ts

```
import { Component, OnInit, Inject } from '@angular/core';
import { PopulationService } from "./population.service";

@Component({
 selector: 'app-india',
 templateUrl: './india.component.html',
 styleUrls: ['./india.component.css']
})
export class IndiaComponent
{
 population: number = 1.32;

 constructor(@Inject(PopulationService) private ps : PopulationService)
 {
 }

 onIncreaseClick()
 {
 this.population++;
 this.ps.next(this.population);
 }
}
```

```
onErrorHandler()
{
 this.ps.error();
}

onCompleteHandler()
{
 this.ps.complete();
}
}
```

c:\angular\app1\src\app\india\india.component.html

```
<div class="class1">
<h4>India</h4>
<input type="button" value="Increase" (click)="onIncreaseClick()">
<input type="button" value="Error" (click)="onErrorHandler()">
<input type="button" value="Complete" (click)="onCompleteHandler()">
</div>
```

c:\angular\app1\src\app\Usa\Usa.component.ts

```
import { Component, OnInit, OnDestroy, Inject } from '@angular/core';
import { PopulationService } from './population.service';
import { Subscription } from "rxjs/Subscription";
import "rxjs/Rx";

@Component({
 selector: 'app-usa',
 templateUrl: './usa.component.html',
 styleUrls: ['./usa.component.css']
})
export class UsaComponent implements OnInit, OnDestroy
{
 indiaPopulation: number = 1.32;
 subscription: Subscription;

 constructor(@Inject(PopulationService) private ps: PopulationService)
 {
 }

 ngOnInit()
 {
 this.subscription = this.ps.myObservable
 .filter((item) =>
 {
 if (parseInt(String(item)) % 2 == 0)
 {
 return true;
 }
 else
 {

```

```
 return false;
 }
})
.subscribe(
(n) =>
{
 this.indiapopulation = n;
},
(error) =>
{
 console.log(error);
},
() =>
{
 console.log("Completed");
});
}

ngOnDestroy()
{
 this.subscription.unsubscribe();
}
}
```

c:\angular\app1\src\app\Usa\Usa.component.html

```
<div class="class1">
<h4>United States</h4>
India population: {{indiapopulation}} million.
</div>
```

#### Executing the application:

- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```

**RxJS - Custom Observables with Services with Filter**

**India**

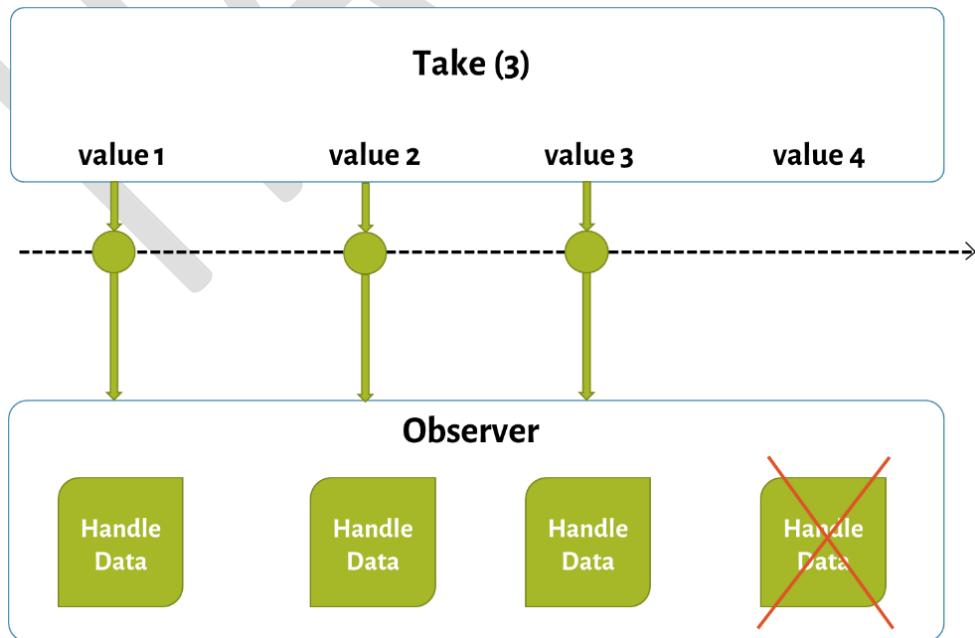
Increase Error Complete

**United States**

India population: 6.32 billion.

### Take

- It executes the emits only the specified no. of data packets maximum to the observer.



## Working with Take

Observablevariable.take( n);

### Take - Example

#### Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g component India
ng g component Usa
ng g service Population
```

#### c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
},
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",

```

```
"@types/jasmine": "~2.8.3",
"@types/jasminewd2": "~2.0.2",
"@types/node": "~6.0.60",
"codelyzer": "^4.0.1",
"jasmine-core": "~2.8.0",
"jasmine-spec-reporter": "~4.2.1",
"karma": "~2.0.0",
"karma-chrome-launcher": "~2.2.0",
"karma-coverage-istanbul-reporter": "^1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\population.service.ts

```
import { Injectable } from '@angular/core';
import { Observable } from "rxjs/Observable";
import { Observer } from "rxjs/Observer";

@Injectable()
export class PopulationService
{
 myObservable: Observable<number>;
 myObserver: Observer<number>;

 constructor()
 {
 this.myObservable = Observable.create((observer: Observer<number>) =>
 {
 this.myObserver = observer;
 });
 }

 next(data)
 {
 this.myObserver.next(data);
 }

 error()
 {
 this.myObserver.error("some error");
 }

 complete()
 {
 this.myObserver.complete();
 }
}
```

c:\angular\app1\src\styles.css

```
.class1
{
 background-color: #91e3e9;
 width: 500px;
 height: 150px;
 margin: 20px;
 float: left;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";
import { IndiaComponent } from './india/india.component';
import { UsaComponent } from './usa/usa.component';
import { PopulationService } from "./population.service";

@NgModule({
 declarations: [
 AppComponent,
 IndiaComponent,
 UsaComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [PopulationService],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div>
<h4>RxJS - Custom Observables with Services with Take</h4>
<app-india></app-india>
<app-usa></app-usa>
</div>
```

c:\angular\app1\src\app\india\india.component.ts

```
import { Component, OnInit, Inject } from '@angular/core';
import { PopulationService } from './population.service';

@Component({
 selector: 'app-india',
 templateUrl: './india.component.html',
 styleUrls: ['./india.component.css']
})
export class IndiaComponent {
 population: number = 1.32;

 constructor(@Inject(PopulationService) private ps : PopulationService)
 {

 }

 onIncreaseClick()
 {
 this.population++;
 this.ps.next(this.population);
 }

 onErrorClick()
 {
 this.ps.error();
 }

 onCompleteClick()
 {
 this.ps.complete();
 }
}
```

c:\angular\app1\src\app\india\india.component.html

```
<div class="class1">
 <h4>India</h4>
 <input type="button" value="Increase" (click)="onIncreaseClick()">
 <input type="button" value="Error" (click)="onErrorClick()">
 <input type="button" value="Complete" (click)="onCompleteClick()">
</div>
```

c:\angular\app1\src\app\Usa\Usa.component.ts

```
import { Component, OnInit, OnDestroy, Inject } from '@angular/core';
import { PopulationService } from './population.service';
import { Subscription } from "rxjs/Subscription";
import "rxjs/Rx";

@Component({
 selector: 'app-usa',
 templateUrl: './usa.component.html',
 styleUrls: ['./usa.component.css']
})
export class UsaComponent implements OnInit, OnDestroy {
 population: number = 1.32;
 subscription: Subscription;

 constructor(private ps: PopulationService) { }

 ngOnInit() {
 this.subscription = this.ps.getPopulation().subscribe(data => {
 this.population = data;
 });
 }

 ngOnDestroy() {
 this.subscription.unsubscribe();
 }

 onIncreaseClick() {
 this.population++;
 this.ps.next(this.population);
 }

 onErrorClick() {
 this.ps.error();
 }

 onCompleteClick() {
 this.ps.complete();
 }
}
```

```
styleUrls: ['./usa.component.css']
})
export class UsaComponent implements OnInit, OnDestroy
{
 indiapopulation: number = 1.32;
 subscription: Subscription;

 constructor(@Inject(PopulationService) private ps : PopulationService)
 {

 }

 ngOnInit()
 {
 this.subscription = this.ps.myObservable
 .take(3)
 .subscribe(
 (n) =>
 {
 this.indiapopulation = n;
 },
 (error) =>
 {
 console.log(error);
 },
 () =>
 {
 console.log("Completed");
 });
 }

 ngOnDestroy()
 {
 this.subscription.unsubscribe();
 }
}
```

c:\angular\app1\src\app\Usa\Usa.component.html

```
<div class="class1">
 <h4>United States</h4>
 India population: {{indiapopulation}} million.
</div>
```

#### Executing the application:

- Open Command Prompt and enter the following commands:  
cd c:\angular\app1  
ng serve
- Open the browser and enter the following URL:  
<http://localhost:4200>

**RxJS - Custom Observables with Services with Take**

**India**

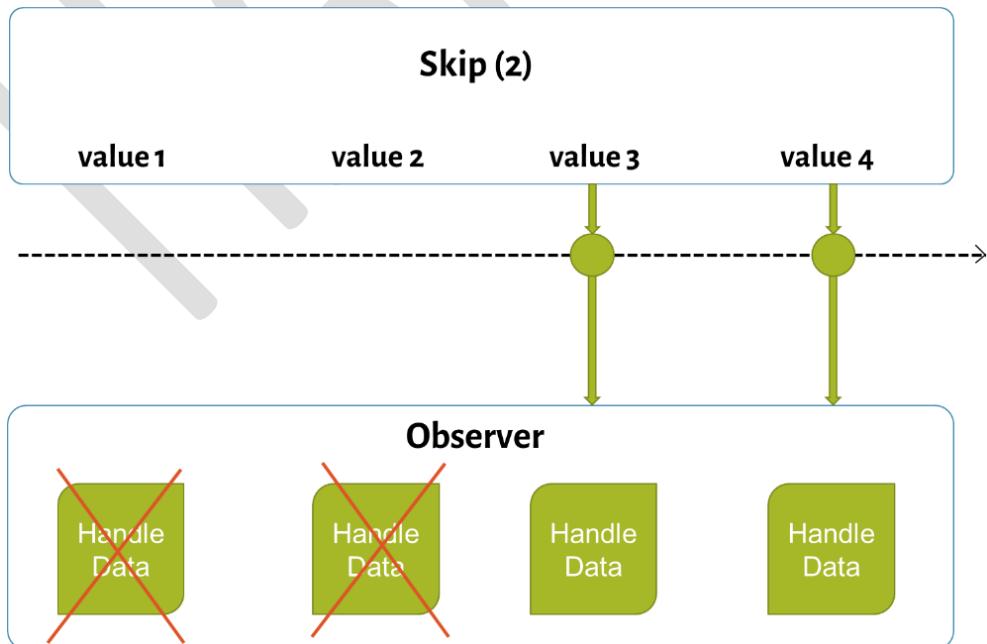
Increase Error Complete

**United States**

India population: 4.32 billion.

### Skip

- It executes the skips the specified no. of data packets, and emits the remaining data packets to the observer.



## Working with Skip

Observablevariable.skip( n);

### Skip - Example

#### Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g component India
ng g component Usa
ng g service Population
```

#### c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
},
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",

```

```
"@types/jasmine": "~2.8.3",
"@types/jasminewd2": "~2.0.2",
"@types/node": "~6.0.60",
"codelyzer": "^4.0.1",
"jasmine-core": "~2.8.0",
"jasmine-spec-reporter": "~4.2.1",
"karma": "~2.0.0",
"karma-chrome-launcher": "~2.2.0",
"karma-coverage-istanbul-reporter": "^1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\population.service.ts

```
import { Injectable } from '@angular/core';
import { Observable } from "rxjs/Observable";
import { Observer } from "rxjs/Observer";

@Injectable()
export class PopulationService
{
 myObservable: Observable<number>;
 myObserver: Observer<number>;

 constructor()
 {
 this.myObservable = Observable.create((observer: Observer<number>) =>
 {
 this.myObserver = observer;
 });
 }

 next(data)
 {
 this.myObserver.next(data);
 }

 error()
 {
 this.myObserver.error("some error");
 }

 complete()
 {
 this.myObserver.complete();
 }
}
```

c:\angular\app1\src\styles.css

```
.class1
{
 background-color: #91e3e9;
 width: 500px;
 height: 150px;
 margin: 20px;
 float: left;
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";
import { IndiaComponent } from './india/india.component';
import { UsaComponent } from './usa/usa.component';
import { PopulationService } from "./population.service";

@NgModule({
 declarations: [
 AppComponent,
 IndiaComponent,
 UsaComponent
],
 imports: [
 BrowserModule, FormsModule
],
 providers: [PopulationService],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{}
```

c:\angular\app1\src\app\app.component.html

```
<div>
<h4>RxJS - Custom Observables with Services with Skip</h4>
```

```
<app-india></app-india>
<app-usa></app-usa>
</div>
```

c:\angular\app1\src\app\india\india.component.ts

```
import { Component, OnInit, Inject } from '@angular/core';
import { PopulationService } from './population.service';

@Component({
 selector: 'app-india',
 templateUrl: './india.component.html',
 styleUrls: ['./india.component.css']
})
export class IndiaComponent {
 population: number = 1.32;

 constructor(@Inject(PopulationService) private ps : PopulationService) {
 }

 onIncreaseClick() {
 this.population++;
 this.ps.next(this.population);
 }

 onErrorClick() {
 this.ps.error();
 }

 onCompleteClick() {
 this.ps.complete();
 }
}
```

c:\angular\app1\src\app\india\india.component.html

```
<div class="class1">
 <h4>India</h4>
 <input type="button" value="Increase" (click)="onIncreaseClick()">
 <input type="button" value="Error" (click)="onErrorHandler()">
 <input type="button" value="Complete" (click)="onCompleteClick()">
</div>
```

c:\angular\app1\src\app\Usa\Usa.component.ts

```
import { Component, OnInit, OnDestroy, Inject } from '@angular/core';
import { PopulationService } from './population.service';
import { Subscription } from "rxjs/Subscription";
import "rxjs/Rx";
```

```
@Component({
 selector: 'app-usa',
 templateUrl: './usa.component.html',
 styleUrls: ['./usa.component.css']
})
export class UsaComponent implements OnInit, OnDestroy
{
 indiapopulation: number = 1.32;
 subscription: Subscription;

 constructor(@Inject(PopulationService) private ps : PopulationService)
 {

 }

 ngOnInit()
 {
 this.subscription = this.ps.myObservable
 .skip(3)
 .subscribe(
 (n) =>
 {
 this.indiapopulation = n;
 },
 (error) =>
 {
 console.log(error);
 },
 () =>
 {
 console.log("Completed");
 });
 }

 ngOnDestroy()
 {
 this.subscription.unsubscribe();
 }
}
```

c:\angular\app1\src\app\Usa\Usa.component.html

```
<div class="class1">
 <h4>United States</h4>
 India population: {{indiapopulation}} million.
</div>
```

#### Executing the application:

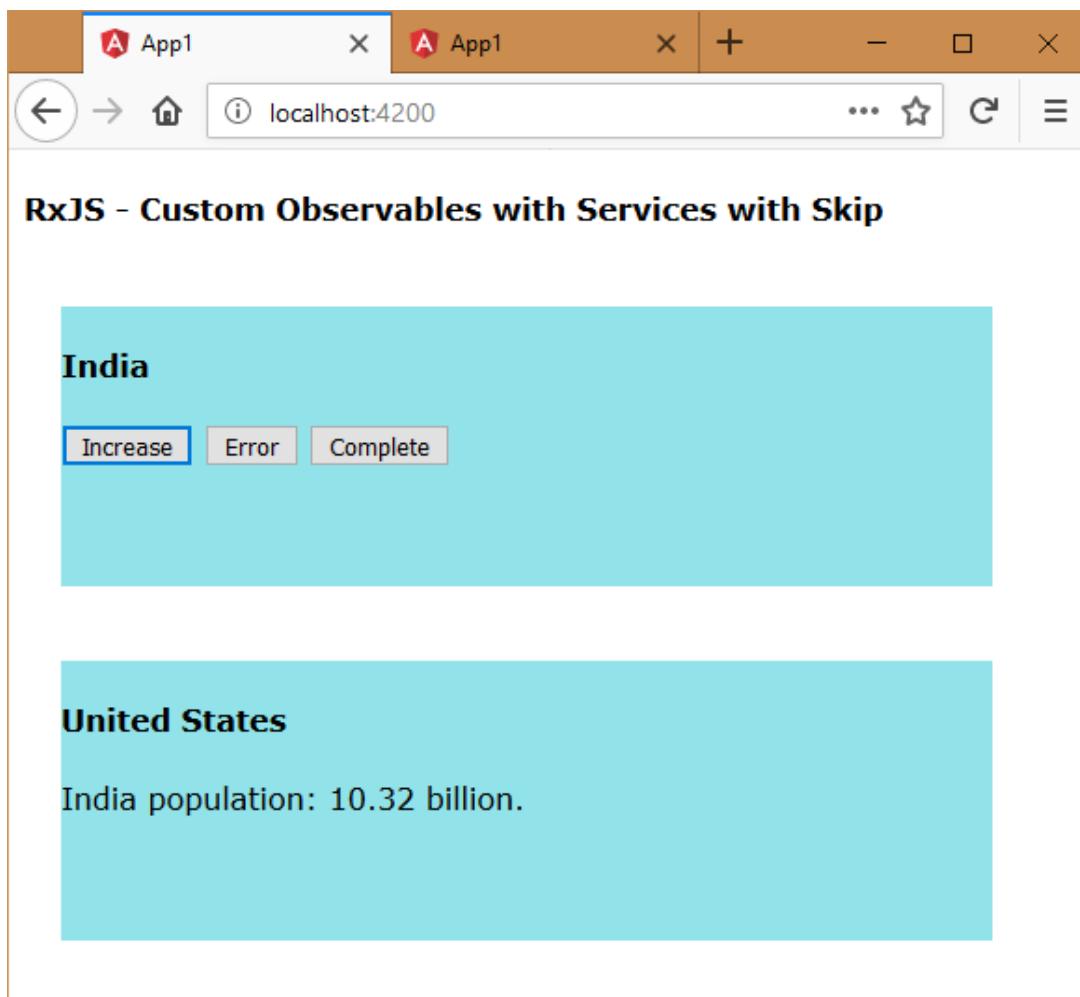
- Open Command Prompt and enter the following commands:

```
cd c:\angular\app1
```

```
ng serve
```

- Open the browser and enter the following URL:

<http://localhost:4200>



### AJAX with Observable - Java - Get

#### Setting-up Environment for Java

- Install Java from "<https://java.com/en/download>".
- Add "C:\Program Files\Java\jdk1.8.0\_172\bin" as "Path" of system variables.
- Add "JAVA\_HOME" with "C:\Program Files\Java\jdk1.8.0\_172" in system variables.
- Download tomcat from "<https://tomcat.apache.org/download-90.cgi>". Click on "zip" in "Core". You will get a file called "apache-tomcat-9.0.7.zip". Right click on "apache-tomcat-9.0.7.zip" and click on "Extract All". Copy all contents of the extracted folder into "c:\tomcat" folder.
- Open Command Prompt and enter the following commands:  
cd c:\tomcat\bin  
startup.bat

---

c:\tomcat\webapps\ROOT\WEB-INF\classes\SampleServlet.java

```

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SampleServlet extends HttpServlet
{
 public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException,IOException
 {
 PrintWriter out = response.getWriter();
 out.println("[{ \"empid\": 1, \"empname\": \"Scott\", \"salary\": 4000 }, { \"empid\": 2, \"empname\":
\"Allen\", \"salary\": 7500 }, { \"empid\": 3, \"empname\": \"Jones\", \"salary\": 9200 }, { \"empid\": 4,
\"empname\": \"James\", \"salary\": 8400 }, { \"empid\": 5, \"empname\": \"Smith\", \"salary\": 5600 }]");
 }
}

```

c:\tomcat\webapps\ROOT\WEB-INF\web.xml

---

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
 http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
 version="4.0"
 metadata-complete="true">

 <display-name>Welcome to Tomcat</display-name>
 <description>
 Welcome to Tomcat
 </description>

 <servlet>
 <servlet-name>SampleServlet</servlet-name>
 <servlet-class>SampleServlet</servlet-class>
 </servlet>

 <servlet-mapping>
 <servlet-name>SampleServlet</servlet-name>
 <url-pattern>/SampleServlet</url-pattern>
 </servlet-mapping>

</web-app>

```

### Creating Application

- Open Command Prompt and enter the following commands:

```

cd c:\angular
ng new app1
cd c:\angular\app1
ng g class Employee

```

```
ng g service Employees
```

c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
 },
 "devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
 }
}
```

---

c:\angular\app1\src\app\employee.ts

```
export class Employee
{
 empid: number;
 empname: string;
 salary: number;

 constructor(a, b, c)
 {
 this.empid = a;
 this.empname = b;
 this.salary = c;
 }
}
```

c:\angular\app1\src\app\employees.service.ts

```
import { Injectable, Inject } from '@angular/core';
import { HttpClient } from "@angular/common/http";
import { Employee } from "./employee";
import { Observable } from 'rxjs/Observable';

@Injectable()
export class EmployeesService
{
 constructor(@Inject(HttpClient) private http: HttpClient)
 {}

 getEmployees(): Observable<Employee[]>
 {
 return this.http.get<Employee[]>("/SampleServlet", { responseType: "json" });
 }
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from "@angular/forms";
import { HttpClientModule } from "@angular/common/http";

import { AppComponent } from './app.component';
import { EmployeesService } from './employees.service';

@NgModule({
 declarations: [
 AppComponent
],
 imports: [
 BrowserModule, FormsModule, HttpClientModule
],
 providers: [EmployeesService],
})
```

```
 bootstrap: [AppComponent]
 })
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component, Inject } from '@angular/core';
import { HttpClient } from "@angular/common/http";
import { Employee } from "./employee";
import { EmployeesService } from "./employees.service";

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{
 employees: Employee[] = [];

 constructor(@Inject(EmployeesService) private employeesService: EmployeesService)
 {
 }

 onGetDataClick()
 {
 this.employeesService.getEmployees().subscribe(this.onAjaxSuccess, this.onAjaxError);
 }

 onAjaxSuccess = (response) =>
 {
 this.employees = response;
 }

 onAjaxError = () =>
 {
 alert("error");
 }
}
```

c:\angular\app1\src\app\app.component.html

```
<div>
<h4>Ajax - Java – Observable - Get</h4>
<input type="button" value="Get Data" (click)="onGetDataClick()">
<table border="1">
 <tr>
 <th>Emp ID</th>
 <th>Emp Name</th>
 <th>Salary</th>
 </tr>
 <tr *ngFor="let employee of employees">
 <td>{{employee.empid}}</td>
```

```

<td>{{employee.empname}}</td>
<td>{{employee.salary}}</td>
</tr>
</table>
</div>

```

### Executing the application:

- Open Command Prompt and enter the following commands:

```
cd c:\tomcat\webapps\ROOT\WEB-INF\classes
```

```
javac -cp c:\tomcat\lib\servlet-api.jar SampleServlet.java
```

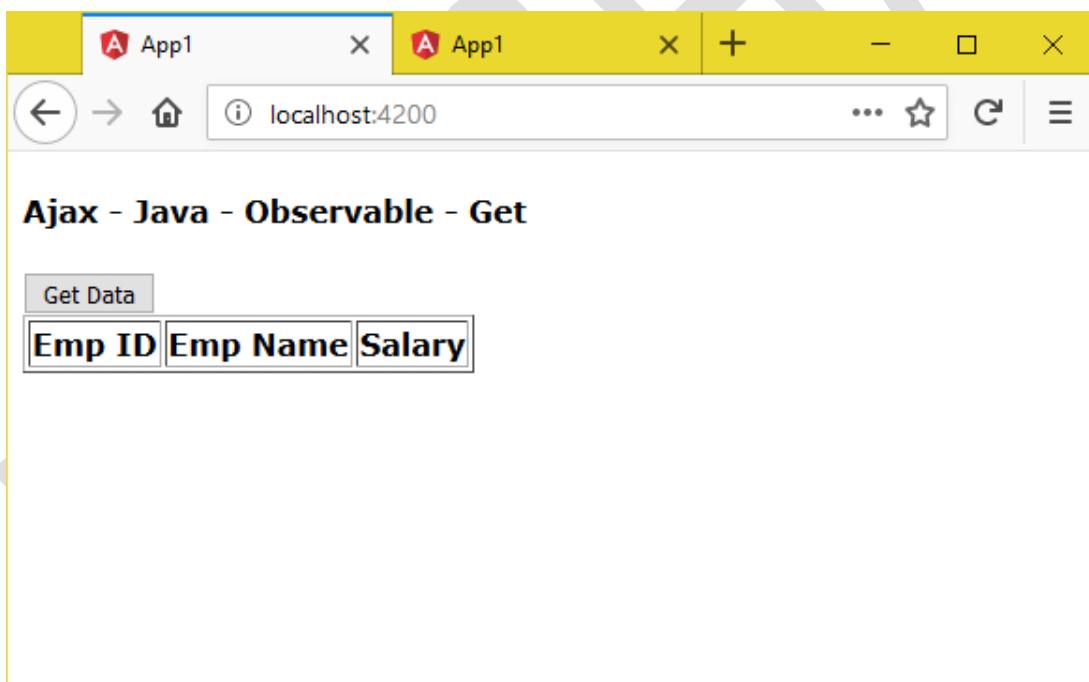
```
cd c:\angular\app1
```

```
ng build --prod
```

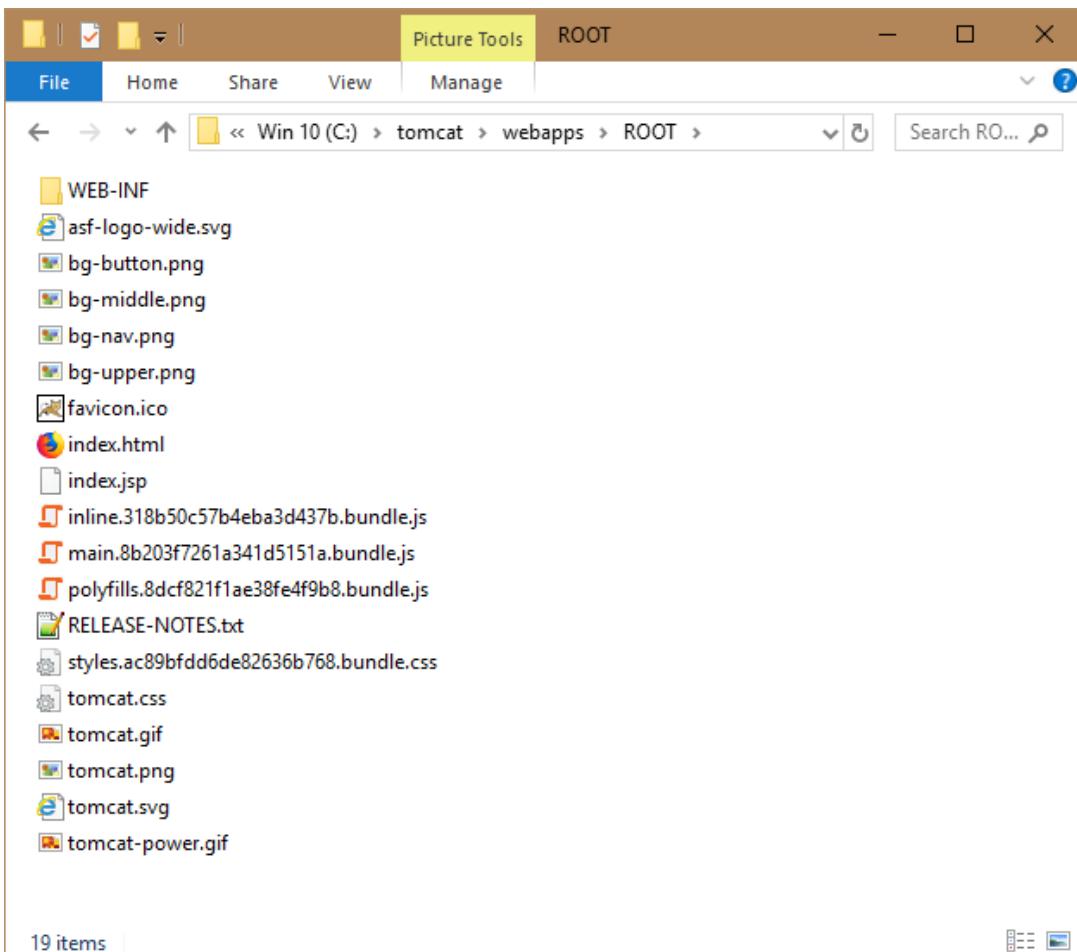
```
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



- Copy all files from "c:\angular\app1\dist" folder to "c:\tomcat\webapps\ROOT".



- Open the browser and enter the following URL:

<http://localhost:8080/index.html>

The screenshot shows a web browser window with the following details:

- Title Bar:** App1
- Address Bar:** localhost:8080
- Content:** Ajax - Java - Observable - Get
- Table:** A data table with the following rows:

| Emp ID | Emp Name | Salary |
|--------|----------|--------|
| 1      | Scott    | 4000   |
| 2      | Allen    | 7500   |
| 3      | Jones    | 9200   |
| 4      | James    | 8400   |
| 5      | Smith    | 5600   |

Click on "Get Data".

## Map HTTP Request - Example

### Setting-up Environment for Java

- Install Java from “<https://java.com/en/download>”.
- Add “C:\Program Files\Java\jdk1.8.0\_172\bin” as “Path” of system variables.
- Add “JAVA\_HOME” with “C:\Program Files\Java\jdk1.8.0\_172” in system variables.
- Download tomcat from “<https://tomcat.apache.org/download-90.cgi>”. Click on “zip” in “Core”. You will get a file called “apache-tomcat-9.0.7.zip”. Right click on “apache-tomcat-9.0.7.zip” and click on “Extract All”. Copy all contents of the extracted folder into “c:\tomcat” folder.
- Open Command Prompt and enter the following commands:  
 cd c:\tomcat\bin  
 startup.bat

c:\tomcat\webapps\ROOT\WEB-INF\classes\SampleServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SampleServlet extends HttpServlet
{
 public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException,IOException
 {
 PrintWriter out = response.getWriter();
 out.println("[{ \"empid\": 1, \"empname\": \"Scott\", \"salary\": 4000 }, { \"empid\": 2, \"empname\":
\"Allen\", \"salary\": 7500 }, { \"empid\": 3, \"empname\": \"Jones\", \"salary\": 9200 }, { \"empid\": 4,
\"empname\": \"James\", \"salary\": 8400 }, { \"empid\": 5, \"empname\": \"Smith\", \"salary\": 5600 }]");
 }
}
```

c:\tomcat\webapps\ROOT\WEB-INF\web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
 http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
 version="4.0"
 metadata-complete="true">

 <display-name>Welcome to Tomcat</display-name>
 <description>
 Welcome to Tomcat
 </description>

 <servlet>
 <servlet-name>SampleServlet</servlet-name>
 <servlet-class>SampleServlet</servlet-class>
 </servlet>
```

```
<servlet-mapping>
 <servlet-name>SampleServlet</servlet-name>
 <url-pattern>/SampleServlet</url-pattern>
</servlet-mapping>

</web-app>
```

## Creating Application

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
cd c:\angular\app1
ng g class Employee
ng g service Employees
```

### c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
 },
 "devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "codelyzer": "3.2.1",
 "jasmine-core": "3.1.0",
 "jasmine-spec-reporter": "4.1.0",
 "karma": "3.0.0",
 "karma-chrome-launcher": "2.2.2",
 "karma-jasmine": "2.0.1",
 "karma-jasmine-html-reporter": "1.1.0",
 "protractor": "5.4.2",
 "ts-node": "7.0.1",
 "tslint": "5.11.0",
 "typescript": "3.1.6"
 }
}
```

```
"@types/jasmine": "~2.8.3",
"@types/jasminewd2": "~2.0.2",
"@types/node": "~6.0.60",
"codelyzer": "^4.0.1",
"jasmine-core": "~2.8.0",
"jasmine-spec-reporter": "~4.2.1",
"karma": "~2.0.0",
"karma-chrome-launcher": "~2.2.0",
"karma-coverage-istanbul-reporter": "^1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~4.1.0",
"tslint": "~5.9.1",
"typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\employee.ts

```
export class Employee
{
 empid: number;
 empname: string;
 salary: number;

 constructor(a, b, c)
 {
 this.empid = a;
 this.empname = b;
 this.salary = c;
 }
}
```

c:\angular\app1\src\app\employees.service.ts

```
import { Injectable, Inject } from '@angular/core';
import { HttpClient } from "@angular/common/http";
import { Employee } from './employee';
import { Observable } from 'rxjs/Observable';

@Injectable()
export class EmployeesService
{
 constructor(@Inject(HttpClient) private http: HttpClient)
 {

 }

 getEmployees(): Observable<Employee[]>
 {
 return this.http.get<Employee[]>("/SampleServlet", { responseType: "json" });
 }
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from "@angular/forms";
import { HttpClientModule } from "@angular/common/http";

import { AppComponent } from './app.component';
import { EmployeesService } from './employees.service';

@NgModule({
 declarations: [
 AppComponent
],
 imports: [
 BrowserModule, FormsModule, HttpClientModule
],
 providers: [EmployeesService],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component, Inject } from '@angular/core';
import { HttpClient } from "@angular/common/http";
import { Employee } from "./employee";
import { EmployeesService } from "./employees.service";
import "rxjs/Rx";

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{
 employees: Employee[] = [];

 constructor(@Inject(EmployeesService) private employeesService: EmployeesService)
 {
 }

 onGetDataClick()
 {
 this.employeesService.getEmployees()
 .map((emps) =>
 {
 for (var i = 0; i < emps.length; i++)
 {
 emps[i].salary = emps[i].salary * 12;
 }
 })
 .subscribe(emps =>
 {
 this.employees = emps;
 });
 }
}
```

```
 }
 return emps;
 })
 .subscribe(this.onAjaxSuccess, this.onAjaxError);
}

onAjaxSuccess = (response) =>
{
 this.employees = response;
}

onAjaxError = () =>
{
 alert("error");
}
```

c:\angular\app1\src\app\app.component.html

```
<div>
<h4>RxJS - Ajax - Java - Map - Get</h4>
<input type="button" value="Get Data" (click)="onGetDataClick()">
<table border="1">
 <tr>
 <th>Emp ID</th>
 <th>Emp Name</th>
 <th>Salary</th>
 </tr>
 <tr *ngFor="let employee of employees">
 <td>{{employee.empid}}</td>
 <td>{{employee.empname}}</td>
 <td>{{employee.salary}}</td>
 </tr>
</table>
</div>
```

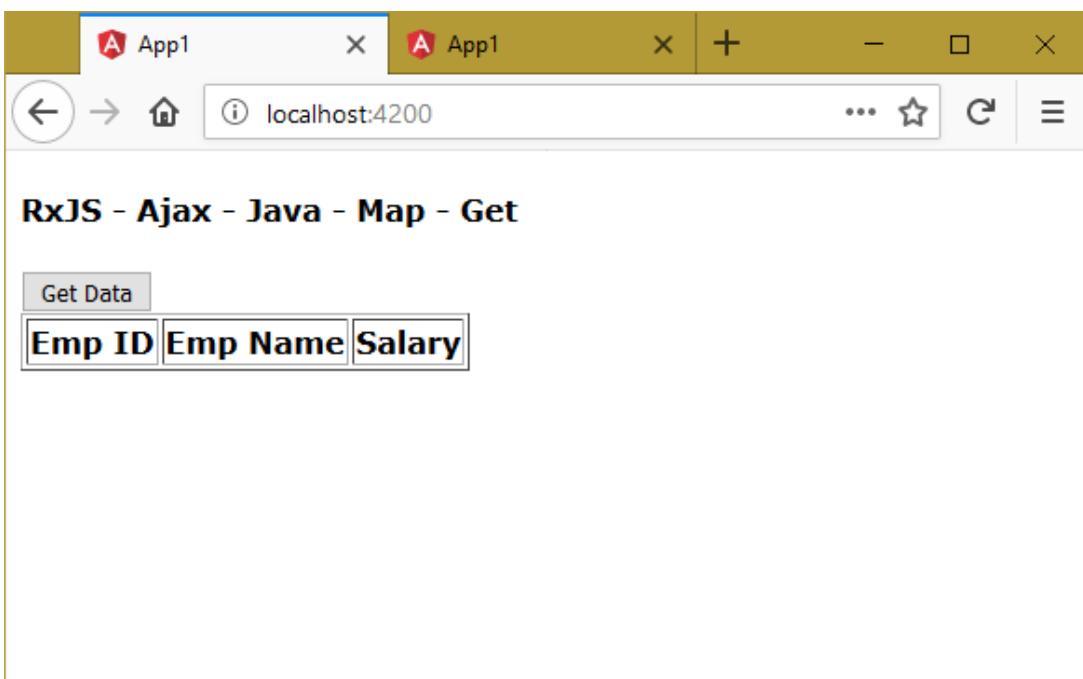
#### Executing the application:

- Open Command Prompt and enter the following commands:

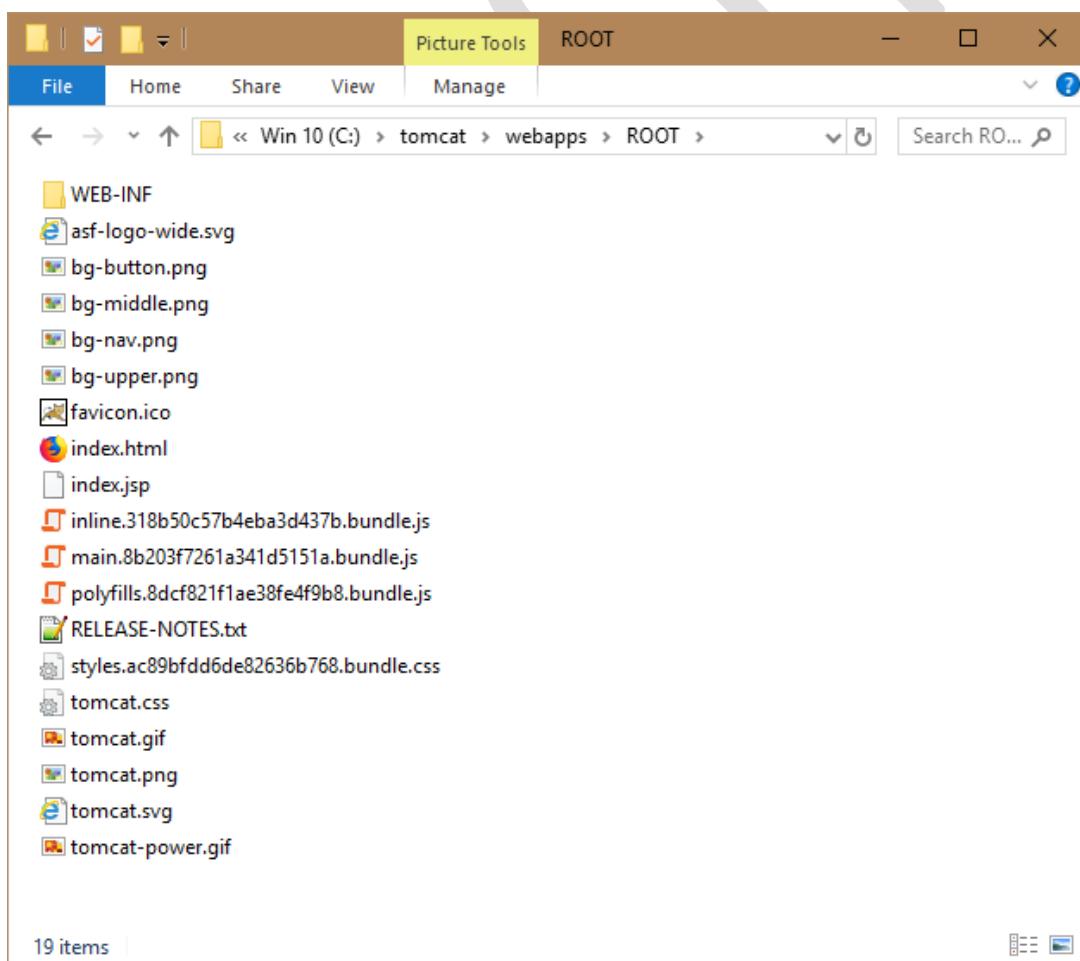
```
cd c:\tomcat\webapps\ROOT\WEB-INF\classes
javac -cp c:\tomcat\lib\servlet-api.jar SampleServlet.java
cd c:\angular\app1
ng build --prod
ng serve
```

- Open the browser and enter the following URL:

```
http://localhost:4200
```



- Copy all files from "c:\angular\app1\dist" folder to "c:\tomcat\webapps\ROOT".



- Open the browser and enter the following URL:

<http://localhost:8080/index.html>

| Emp ID | Emp Name | Salary |
|--------|----------|--------|
| 1      | Scott    | 48000  |
| 2      | Allen    | 90000  |
| 3      | Jones    | 110400 |
| 4      | James    | 100800 |
| 5      | Smith    | 67200  |

Click on "Get Data".

### Cancelling HTTP Request - Example

#### Setting-up Environment for Java

- Install Java from "<https://java.com/en/download>".
- Add "C:\Program Files\Java\jdk1.8.0\_172\bin" as "Path" of system variables.
- Add "JAVA\_HOME" with "C:\Program Files\Java\jdk1.8.0\_172" in system variables.
- Download tomcat from "<https://tomcat.apache.org/download-90.cgi>". Click on "zip" in "Core". You will get a file called "apache-tomcat-9.0.7.zip". Right click on "apache-tomcat-9.0.7.zip" and click on "Extract All". Copy all contents of the extracted folder into "c:\tomcat" folder.
- Open Command Prompt and enter the following commands:  
cd c:\tomcat\bin  
startup.bat

c:\tomcat\webapps\ROOT\WEB-INF\classes\SampleServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SampleServlet extends HttpServlet
{
 public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
 {
 PrintWriter out = response.getWriter();
 }
}
```

```
try
{
 delay();
 out.println("[{"empid": 1, "empname": "Scott", "salary": 4000 }, {"empid": 2,
"empname": "Allen", "salary": 7500 }, {"empid": 3, "empname": "Jones", "salary": 9200 }, {
"empid": 4, "empname": "James", "salary": 8400 }, {"empid": 5, "empname": "Smith", "salary": 5600 }]");
}
catch(Exception e)
{
 out.println("[]");
}
}

public static void delay() throws InterruptedException
{
 Thread.sleep(6000);
}
```

c:\tomcat\webapps\ROOT\WEB-INF\web.xml

---

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
 http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
 version="4.0"
 metadata-complete="true">

 <display-name>Welcome to Tomcat</display-name>
 <description>
 Welcome to Tomcat
 </description>

 <servlet>
 <servlet-name>SampleServlet</servlet-name>
 <servlet-class>SampleServlet</servlet-class>
 </servlet>

 <servlet-mapping>
 <servlet-name>SampleServlet</servlet-name>
 <url-pattern>/SampleServlet</url-pattern>
 </servlet-mapping>

</web-app>
```

## Creating Application

---

- Open Command Prompt and enter the following commands:

```
cd c:\angular
ng new app1
```

```
cd c:\angular\app1
ng g class Employee
ng g service Employees
```

**c:\angular\app1\package.json**

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
 },
 "devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 }
}
```

```
 "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\employee.ts

```
export class Employee
{
 empid: number;
 empname: string;
 salary: number;

 constructor(a, b, c)
 {
 this.empid = a;
 this.empname = b;
 this.salary = c;
 }
}
```

c:\angular\app1\src\app\employees.service.ts

```
import { Injectable, Inject } from '@angular/core';
import { HttpClient } from "@angular/common/http";
import { Employee } from "./employee";
import { Observable } from 'rxjs/Observable';

@Injectable()
export class EmployeesService
{
 constructor(@Inject(HttpClient) private http: HttpClient)
 {

 }

 getEmployees(): Observable<Employee[]>
 {
 return this.http.get<Employee[]>('/SampleServlet', { responseType: "json" });
 }
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from "@angular/forms";
import { HttpClientModule } from "@angular/common/http";

import { AppComponent } from './app.component';
import { EmployeesService } from './employees.service';

@NgModule({
 declarations: [
 AppComponent
],

```

```
imports: [
 BrowserModule, FormsModule, HttpClientModule
],
providers: [EmployeesService],
bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component, Inject } from '@angular/core';
import { HttpClient } from "@angular/common/http";
import { Employee } from "./employee";
import { EmployeesService } from "./employees.service";
import "rxjs/Rx";
import { Subscription } from 'rxjs/Subscription';

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent {
 employees: Employee[] = [];
 showloading: boolean = false;
 subscription: Subscription;

 constructor(@Inject(EmployeesService) private employeesService: EmployeesService)
 {
 }

 onGetDataClick()
 {
 this.showloading = true;
 this.subscription = this.employeesService.getEmployees().subscribe(this.onAjaxSuccess,
 this.onAjaxError);
 }

 onAjaxSuccess = (response) =>
 {
 this.employees = response;
 this.showloading = false;
 }

 onAjaxError = () =>
 {
 this.showloading = false;
 alert("error");
 }

 onCancelClick()
 {
```

```
this.subscription.unsubscribe();
this.showloading = false;
}
}
```

c:\angular\app1\src\app\app.component.html

```
<div>
<h4>RxJS - Cancelling HTTP - Java - Get</h4>
<input type="button" value="Get Data" (click)="onGetDataClick()" *ngIf="!showloading">

<input type="button" value="Cancel" *ngIf="showloading" (click)="onCancelClick()">
<table border="1">
 <tr>
 <th>Emp ID</th>
 <th>Emp Name</th>
 <th>Salary</th>
 </tr>
 <tr *ngFor="let employee of employees">
 <td>{{employee.empid}}</td>
 <td>{{employee.empname}}</td>
 <td>{{employee.salary}}</td>
 </tr>
</table>
</div>
```

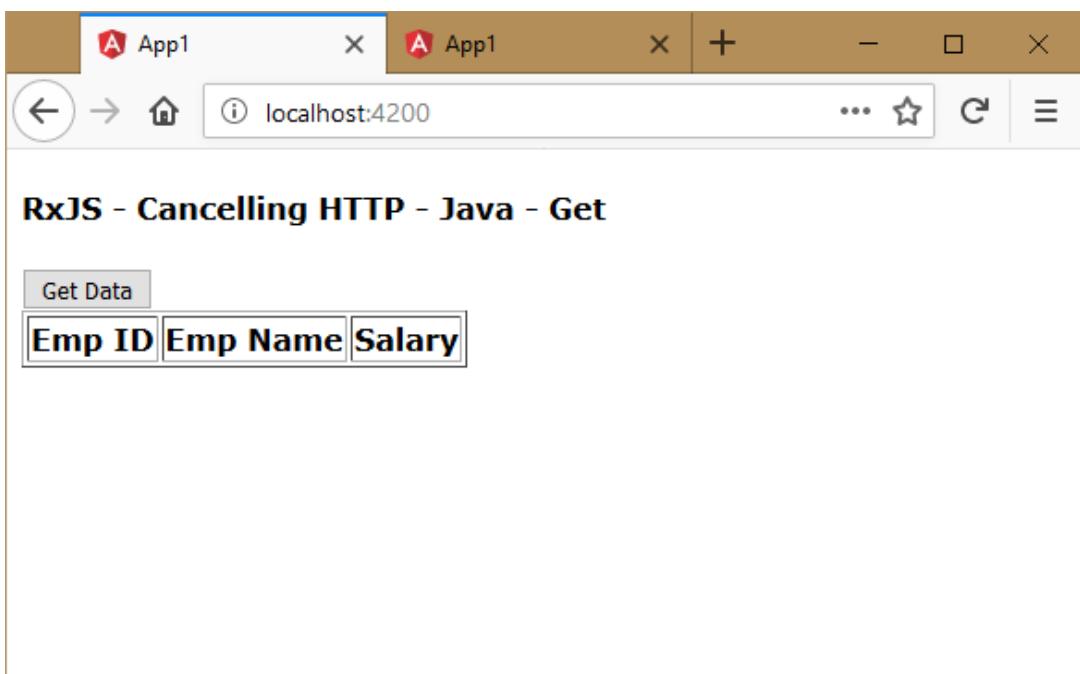
#### Executing the application:

- Place “ajax-loader.gif” file in “src\assets” folder.
- Open Command Prompt and enter the following commands:

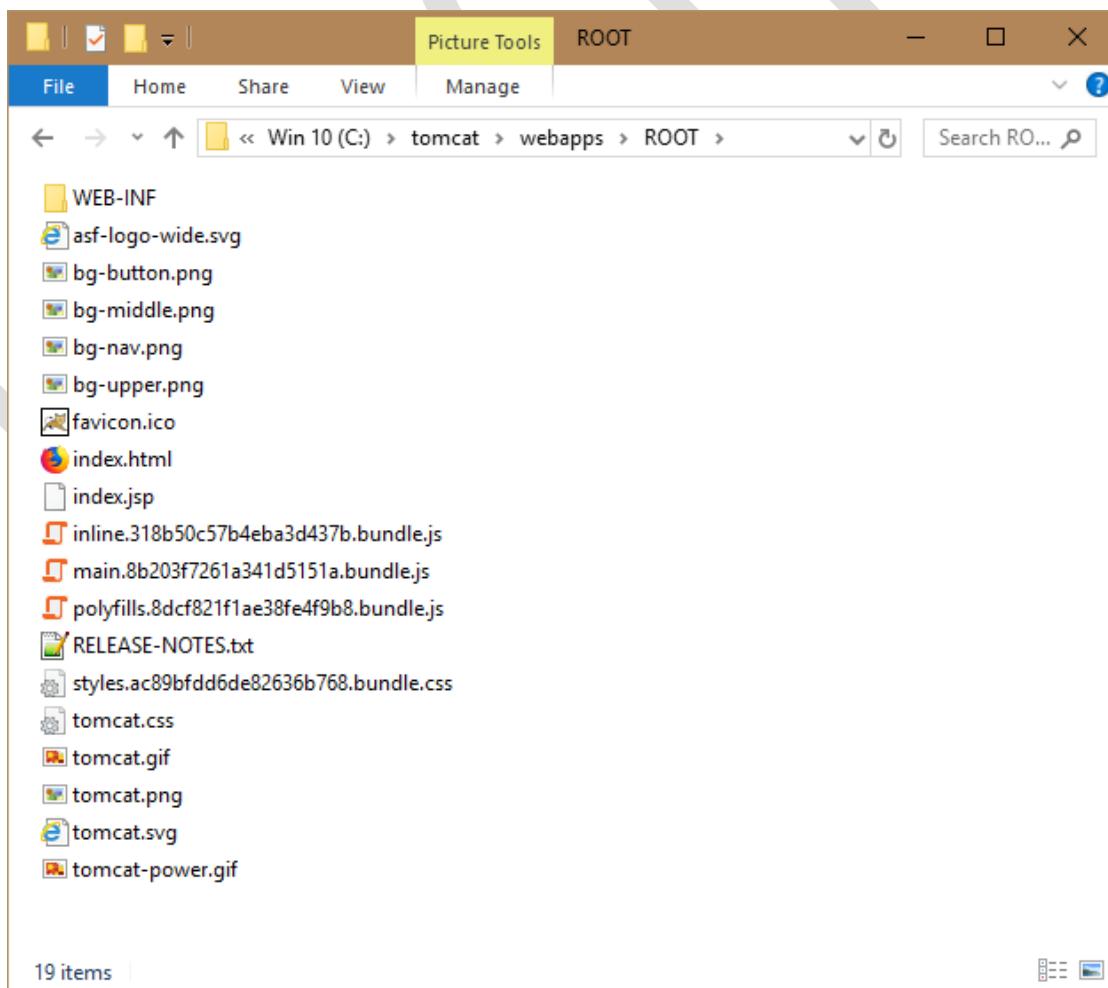
```
cd c:\tomcat\webapps\ROOT\WEB-INF\classes
javac -cp c:\tomcat\lib\servlet-api.jar SampleServlet.java
cd c:\angular\app1
ng build --prod
ng serve
```

- Open the browser and enter the following URL:

<http://localhost:4200>

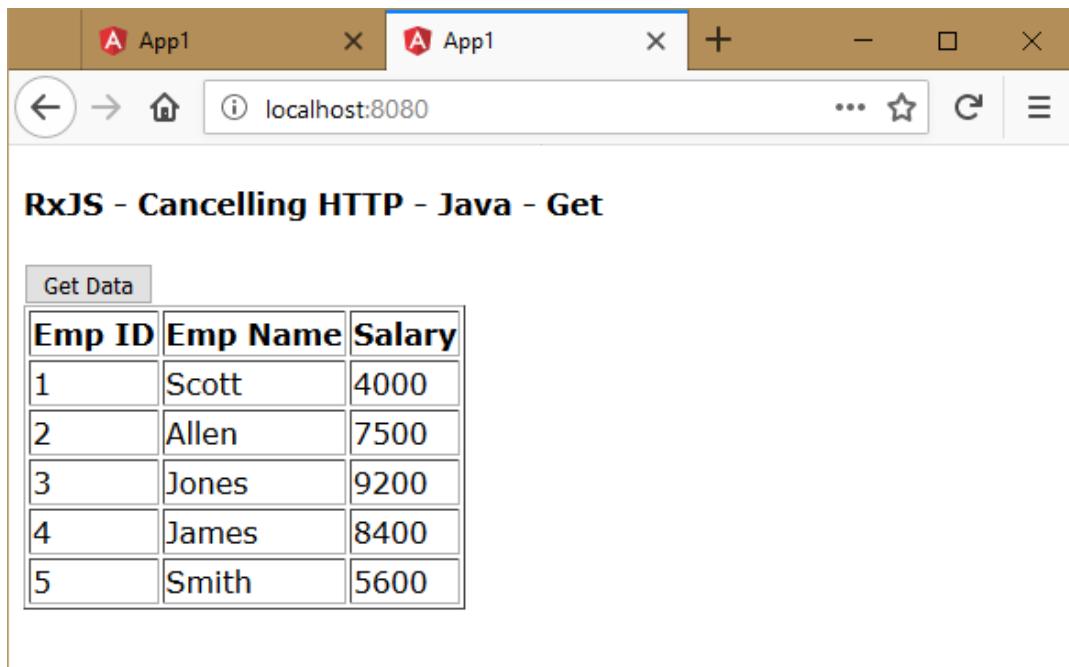


- Copy all files from "c:\angular\app1\dist" folder to "c:\tomcat\webapps\ROOT".



- Open the browser and enter the following URL:

<http://localhost:8080/index.html>



Click on "Get Data".

### Retrying HTTP Request - Example

#### Setting-up Environment for Java

- Install Java from "<https://java.com/en/download>".
- Add "C:\Program Files\Java\jdk1.8.0\_172\bin" as "Path" of system variables.
- Add "JAVA\_HOME" with "C:\Program Files\Java\jdk1.8.0\_172" in system variables.
- Download tomcat from "<https://tomcat.apache.org/download-90.cgi>". Click on "zip" in "Core". You will get a file called "apache-tomcat-9.0.7.zip". Right click on "apache-tomcat-9.0.7.zip" and click on "Extract All". Copy all contents of the extracted folder into "c:\tomcat" folder.
- Open Command Prompt and enter the following commands:  
cd c:\tomcat\bin  
startup.bat

c:\tomcat\webapps\ROOT\WEB-INF\classes\SampleServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SampleServlet extends HttpServlet
{
 public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException,IOException
 {
 if (1 == 1)
```

```

 throw new IOException("some error");
 PrintWriter out = response.getWriter();
 out.println("[{ \"empid\": 1, \"empname\": \"Scott\", \"salary\": 4000 }, { \"empid\": 2, \"empname\":\n\"Allen\", \"salary\": 7500 }, { \"empid\": 3, \"empname\": \"Jones\", \"salary\": 9200 }, { \"empid\": 4,\n\"empname\": \"James\", \"salary\": 8400 }, { \"empid\": 5, \"empname\": \"Smith\", \"salary\": 5600 }]");
}
}

```

c:\tomcat\webapps\ROOT\WEB-INF\web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
 http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
 version="4.0"
 metadata-complete="true">

 <display-name>Welcome to Tomcat</display-name>
 <description>
 Welcome to Tomcat
 </description>

 <servlet>
 <servlet-name>SampleServlet</servlet-name>
 <servlet-class>SampleServlet</servlet-class>
 </servlet>

 <servlet-mapping>
 <servlet-name>SampleServlet</servlet-name>
 <url-pattern>/SampleServlet</url-pattern>
 </servlet-mapping>

</web-app>

```

## Creating Application

- Open Command Prompt and enter the following commands:

```

cd c:\angular
ng new app1
cd c:\angular\app1
ng g class Employee
ng g service Employees

```

c:\angular\app1\package.json

```
{
 "name": "app1",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {

```

```
"ng": "ng",
"start": "ng serve",
"build": "ng build --prod",
"test": "ng test",
"lint": "ng lint",
"e2e": "ng e2e"
},
"private": true,
"dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 "@angular/core": "^5.2.0",
 "@angular/forms": "^5.2.0",
 "@angular/http": "^5.2.0",
 "@angular/platform-browser": "^5.2.0",
 "@angular/platform-browser-dynamic": "^5.2.0",
 "@angular/router": "^5.2.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "^0.8.19"
},
"devDependencies": {
 "@angular/cli": "~1.7.4",
 "@angular/compiler-cli": "^5.2.0",
 "@angular/language-service": "^5.2.0",
 "@types/jasmine": "~2.8.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.8.0",
 "jasmine-spec-reporter": "~4.2.1",
 "karma": "~2.0.0",
 "karma-chrome-launcher": "~2.2.0",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~4.1.0",
 "tslint": "~5.9.1",
 "typescript": "~2.5.3"
}
}
```

c:\angular\app1\src\app\employee.ts

```
export class Employee
{
 empid: number;
 empname: string;
 salary: number;

 constructor(a, b, c)
```

```
{
 this.empid = a;
 this.empname = b;
 this.salary = c;
}
}
```

c:\angular\app1\src\app\employees.service.ts

```
import { Injectable, Inject } from '@angular/core';
import { HttpClient } from "@angular/common/http";
import { Employee } from './employee';
import { Observable } from 'rxjs/Observable';
import "rxjs/Rx";

@Injectable()
export class EmployeesService
{
 constructor(@Inject(HttpClient) private http: HttpClient)
 {}
 {}

 getEmployees(): Observable<Employee[]>
 {}
 return this.http.get<Employee[]>("/SampleServlet", { responseType: "json" }).retry(3);
}
}
```

c:\angular\app1\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from "@angular/forms";
import { HttpClientModule } from "@angular/common/http";

import { AppComponent } from './app.component';
import { EmployeesService } from './employees.service';

@NgModule({
 declarations: [
 AppComponent
],
 imports: [
 BrowserModule, FormsModule, HttpClientModule
],
 providers: [EmployeesService],
 bootstrap: [AppComponent]
})
export class AppModule {}
```

c:\angular\app1\src\app\app.component.ts

```
import { Component, Inject } from '@angular/core';
import { HttpClient } from "@angular/common/http";
```

```

import { Employee } from "./employee";
import { EmployeesService } from "./employees.service";

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent
{
 employees: Employee[] = [];

 constructor(@Inject(EmployeesService) private employeesService: EmployeesService)
 {
 }

 onGetDataClick()
 {
 this.employeesService.getEmployees().subscribe(this.onAjaxSuccess, this.onAjaxError);
 }

 onAjaxSuccess = (response) =>
 {
 this.employees = response;
 }

 onAjaxError = () =>
 {
 alert("error");
 }
}

```

c:\angular\app1\src\app\app.component.html

```

<div>
 <h4>RxJS - Retrying Http Request - Java - Get</h4>
 <input type="button" value="Get Data" (click)="onGetDataClick()">
 <table border="1">
 <tr>
 <th>Emp ID</th>
 <th>Emp Name</th>
 <th>Salary</th>
 </tr>
 <tr *ngFor="let employee of employees">
 <td>{{employee.empid}}</td>
 <td>{{employee.empname}}</td>
 <td>{{employee.salary}}</td>
 </tr>
 </table>
</div>

```

### Executing the application:

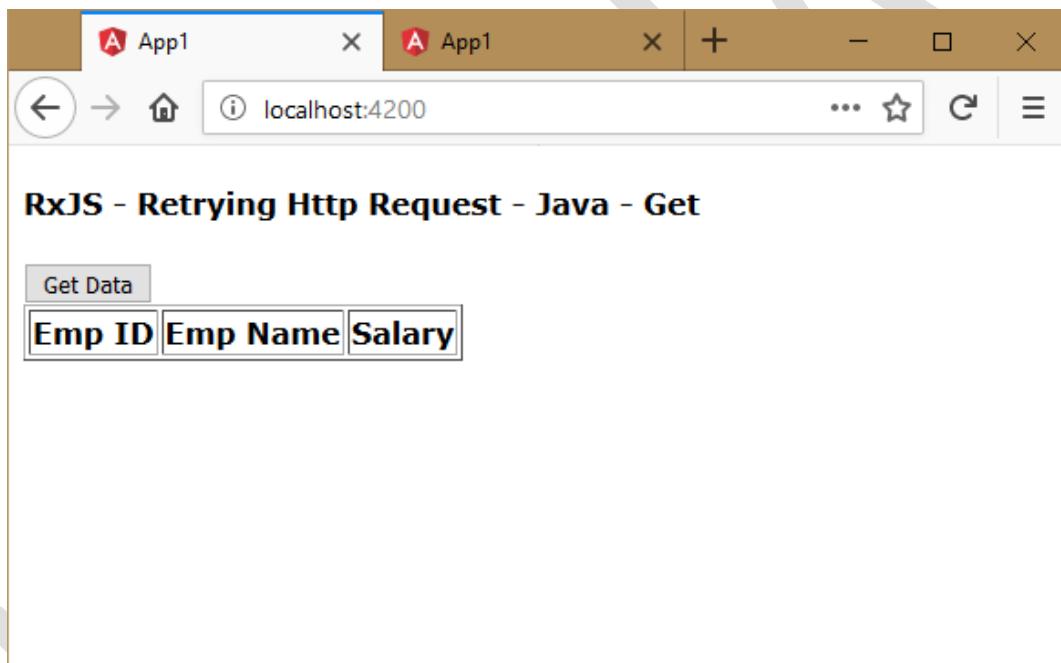
- Place “ajax-loader.gif” file in “src\assets” folder.
- Open Command Prompt and enter the following commands:

```
cd c:\tomcat\webapps\ROOT\WEB-INF\classes
javac -cp c:\tomcat\lib\servlet-api.jar SampleServlet.java

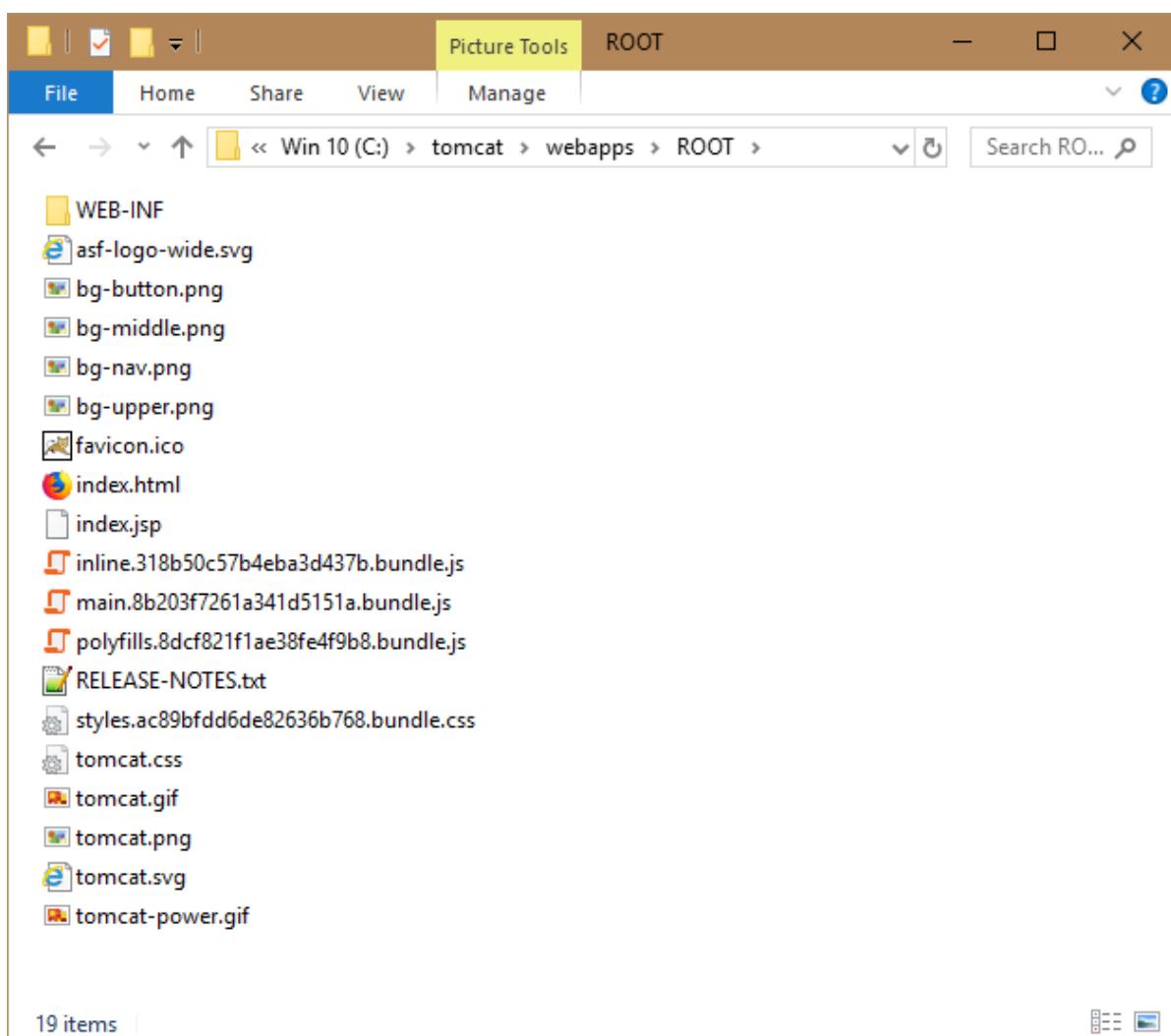
cd c:\angular\app1

ng build --prod

ng serve
```
- Open the browser and enter the following URL:  
<http://localhost:4200>



- Copy all files from “c:\angular\app1\dist” folder to “c:\tomcat\webapps\ROOT”.



- Open the browser and enter the following URL:  
<http://localhost:8080/index.html>

The screenshot shows a browser window with two tabs: 'App1' and 'App1'. The active tab displays the title 'RxJS - Retrying Http Request - Java - Get'. Below the title is a table with three columns: 'Emp ID', 'Emp Name', and 'Salary'. A blue-bordered button labeled 'Get Data' is positioned above the table. The table currently contains no data. At the bottom of the page, there is a network monitoring tool showing four requests to 'SampleServlet' with a status of 500. The requests are listed as follows:

| Sta... | Me... | File          | Domain    | Ca... | Type | Tra...  | Size    |
|--------|-------|---------------|-----------|-------|------|---------|---------|
| ◆ 500  | GET   | SampleServlet | localhost | xhr   | html | 1.59 KB | 1.44 KB |
| ◆ 500  | GET   | SampleServlet | localhost | xhr   | html | 1.59 KB | 1.44 KB |
| ◆ 500  | GET   | SampleServlet | localhost | xhr   | html | 1.59 KB | 1.44 KB |
| ◆ 500  | GET   | SampleServlet | localhost | xhr   | html | 1.59 KB | 1.44 KB |

At the bottom of the network tab, it says '4 requests | 5.75 KB / 6.36 KB transferred | Finish: 20 ms'.

Click on "Get Data".

### Unit Testing

- Unit testing is a process of testing the individual component, whether it is working correctly or not.
- We use “Jasmine” and “Karma” for the unit testing.
- **Jasmine:** Used to create test cases.
- **Protractor:** Used to execute the test cases in single browser.
- **Karma:** Used to execute the test cases in multiple browsers.

### Syntax of test case:

```
import { TestBed, async } from '@angular/core/testing';
import { ComponentClassname } from './filename';

describe("Heading here", () => {
 beforeEach(async() => {
```

```

 TestBed.configureTestingModule({
 declarations: [Componentclassname],
 }).compileComponents();
}));

it("test case name here", async(() => {
 var comp = TestBed.createComponent(Componentclassname).debugElement.componentInstance;
 app.property
 app.method();
 expect(app.property).toBe(value);
 }));
});

```

### Unit Testing - Example

#### Steps

- Command Prompt

```

cd c:\angular
ng new myapp
cd c:\angular\myapp

```

c:\angular\myapp\package.json

```
{
 "name": "myapp",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.0.0",
 "@angular/common": "^5.0.0",
 "@angular/compiler": "^5.0.0",
 "@angular/core": "^5.0.0",
 "@angular/forms": "^5.0.0",
 "@angular/http": "^5.0.0",
 "@angular/platform-browser": "^5.0.0",
 "@angular/platform-browser-dynamic": "^5.0.0",
 "@angular/router": "^5.0.0",
 "core-js": "^2.4.1",
 }
}
```

```
"rxjs": "^5.5.2",
"zone.js": "^0.8.14"
},
"devDependencies": {
 "@angular/cli": "1.6.3",
 "@angular/compiler-cli": "^5.0.0",
 "@angular/language-service": "^5.0.0",
 "@types/jasmine": "~2.5.53",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.6.2",
 "jasmine-spec-reporter": "~4.1.0",
 "karma": "~1.7.0",
 "karma-chrome-launcher": "~2.1.1",
 "karma-cli": "~1.0.1",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 "karma-jasmine-html-reporter": "^0.2.2",
 "protractor": "~5.1.2",
 "ts-node": "~3.2.0",
 "tslint": "~5.7.0",
 "typescript": "~2.4.2"
}
}
```

c:\angular\myapp\src\app\app.component.ts

```
import { Component } from "@angular/core";

@Component({ selector: "app", templateUrl: "./app.component.html" })
export class AppComponent
{
 username: string = null;
 password: string = null;
 msg: string = null;

 CheckLogin()
 {
 if (this.username == "admin" && this.password == "manager")
 {
 this.msg = "Successful login";
 }
 else
 {
 this.msg = "Invalid login";
 }
 }
}
```

```
}
```

c:\angular\myapp\src\app\app.component.html

```
<div>
<form>
<h1>Login</h1>
Username: <input type="text" [(ngModel)]="username" name="username">

Password: <input type="password" [(ngModel)]="password" name="password">

<input type="submit" value="Login" (click)="CheckLogin()">

{{msg}}
</form>
</div>
```

c:\angular\myapp\src\app\app.component.css

```
/* You can add css code of "app component" here */
```

c:\angular\myapp\src\app\app.component.spec.ts

```
import { TestBed, async } from '@angular/core/testing';
import { AppComponent } from './app.component';
import { FormsModule } from "@angular/forms";

describe('AppComponent', () => {
 beforeEach(async(() => {
 TestBed.configureTestingModule({
 declarations: [AppComponent],
 imports: [FormsModule]
 }).compileComponents();
 }));

 it("Login - Success", async(() => {
 const fixture = TestBed.createComponent(AppComponent);
 const app = fixture.debugElement.componentInstance;
 app.username = "admin";
 app.password = "manager";
 app.CheckLogin();
 expect(app.msg).toBe("Successful login");
 }));

 it("Login - Fail", async(() =>
 {
 const fixture = TestBed.createComponent(AppComponent);
 const app = fixture.debugElement.componentInstance;
 app.username = "abc";
 app.password = "def";
 app.CheckLogin();
 expect(app.msg).toBe("Invalid login");
 }));
});
```

```
});
```

c:\angular\myapp\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from "@angular/forms";
import { AppComponent } from './app.component';

@NgModule({
 declarations: [AppComponent],
 imports: [BrowserModule, FormsModule],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule
{}
```

c:\angular\myapp\src\styles.css

```
/* You can add global styles to this file, and also import other style files */
```

c:\angular\myapp\src\main.ts

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production)
{
 enableProdMode();
}
platformBrowserDynamic().bootstrapModule(AppModule)
 .catch(err => console.log(err));
```

c:\angular\myapp\src\index.html

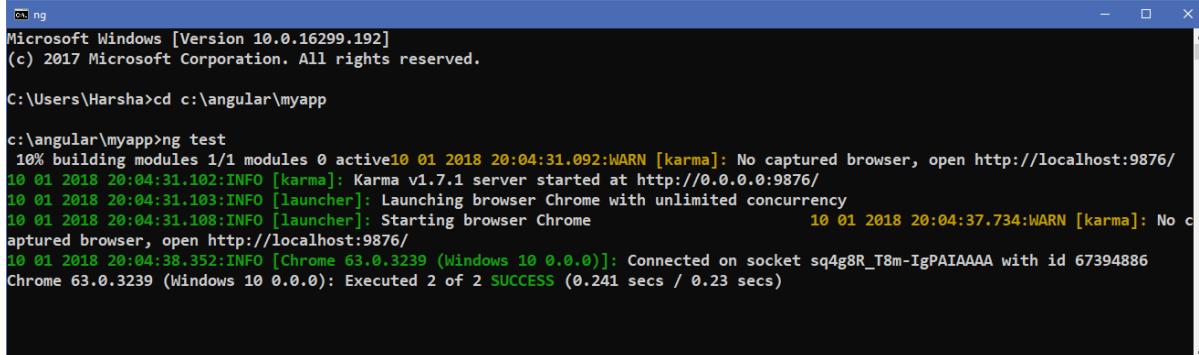
```
<!doctype html>
<html lang="en">
<head>
 <meta charset="utf-8">
 <title>Myapp</title>
 <base href="/">
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
 <app-root></app-root>
</body>
```

</html>

### Executing the application:

- Open Command Prompt and enter the following commands:

```
cd c:\angular\myapp
ng test
```



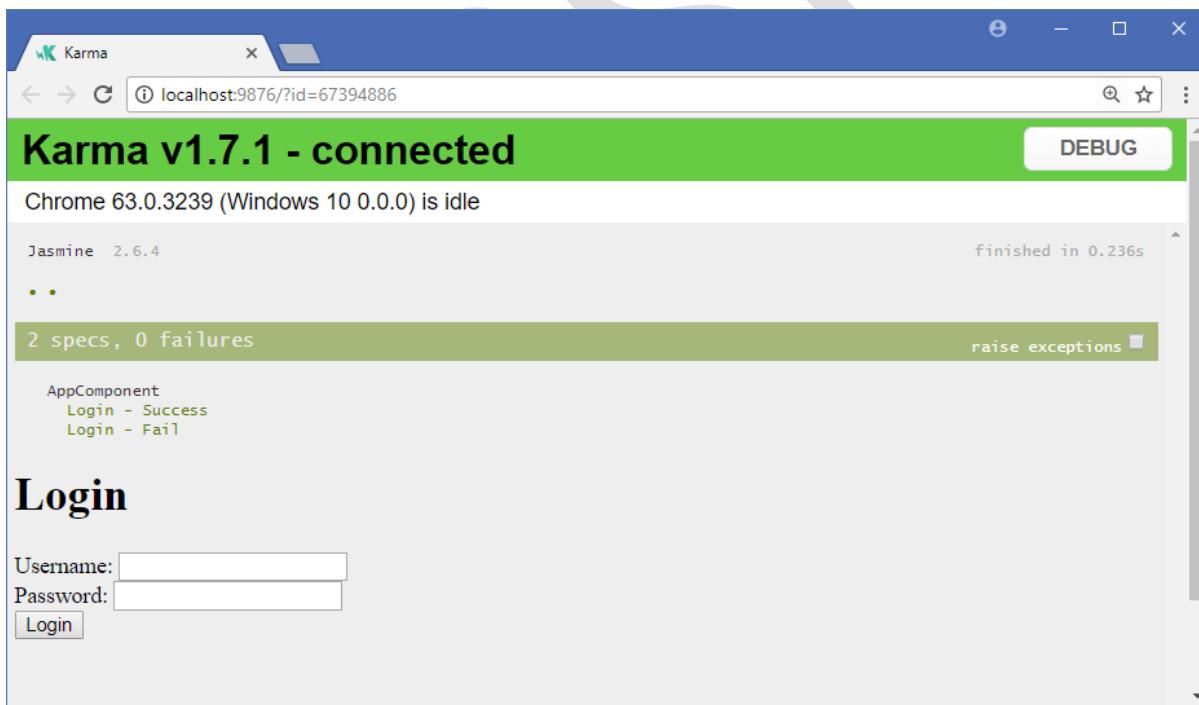
```
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Harsha>cd c:\angular\myapp

c:\angular\myapp>ng test
10% building modules 1/1 modules 0 active
10 01 2018 20:04:31.092:WARN [karma]: No captured browser, open http://localhost:9876/
10 01 2018 20:04:31.102:INFO [karma]: Karma v1.7.1 server started at http://0.0.0.0:9876/
10 01 2018 20:04:31.103:INFO [launcher]: Launching browser Chrome with unlimited concurrency
10 01 2018 20:04:31.108:INFO [launcher]: Starting browser Chrome 10 01 2018 20:04:37.734:WARN [karma]: No captured browser, open http://localhost:9876/
10 01 2018 20:04:38.352:INFO [Chrome 63.0.3239 (Windows 10 0.0.0)]: Connected on socket sq4g8R_T8m-IgPAIAAAA with id 67394886
Chrome 63.0.3239 (Windows 10 0.0.0): Executed 2 of 2 SUCCESS (0.241 secs / 0.23 secs)
```

- It automatically opens the Chrome browser at the following URL:

<http://localhost:9876?id=57394886>



### Animations

- Animation is a process of changing css property value gradually based on the time limit. That means within the specified time limit, the property will change from “start value” to “end value”.
- Angular 2+ animations are used to invoke the css animations programmatically through the component.

**Steps for working with Animations:**

- **Import “@angular/animations” package in “package.json” file:**

```
"@angular/animations": "5.0.0",
```

- **Import “BrowserAnimationsModule” in “app.module.ts”:**

```
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
```

- **Import “BrowserAnimationsModule” into “AppModule”:**

```
@NgModule({ ... , imports: [... , BrowserAnimationsModule] })
class AppModule
{
}
```

- **Import “trigger”, “state”, “style”, “transition”, “animate” in “app.component.ts”:**

```
import { trigger, state, style, transition, animate } from '@angular/animations';
```

- **Define animation:**

```
var myanimations = [
 trigger("animationname", [
 state("state1", style({ cssproperty: "value" })),
 state("state2", style({ cssproperty: "value" })),
 transition("state1 => state2", animate("milli seconds")),
 transition("state2 => state1", animate("milli seconds"))
])
];
```

- **Import animation in the component:**

```
@Component({ ... , animations: myanimations })
class AppComponent
{
}
```

- **Define property in the component:**

```
@Component({ ... , animations: myanimations })
class AppComponent
{
 propertyname : datatype = "state1";
}
```

- **Map “property” of the component to “animation”:**

```
<tag [@animationname]="propertyname">
 ...
```

</tag>

- **Change the value of the “property” of the component:**

```
@Component({ ... , animations: myanimations })
class AppComponent
{
 propertynome : datatype = "state1";

 method()
 {
 this.propertynome = "state2";
 }
}
```

### Animations - Example

#### Steps

- Command Prompt
  - npm install @angular/cli -g
  - cd c:\angular
  - ng new myapp
  - cd c:\angular\myapp
  - ng serve

#### c:\angular\myapp\package.json

```
{
 "name": "myapp",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/animations": "^5.0.0",
 "@angular/common": "^5.0.0",
 "@angular/compiler": "^5.0.0",
 "@angular/core": "^5.0.0",
 "@angular/forms": "^5.0.0",
 "@angular/http": "^5.0.0",
 "@angular/platform-browser": "^5.0.0",
 }
}
```

```

"@angular/platform-browser-dynamic": "^5.0.0",
"@angular/router": "^5.0.0",
"core-js": "^2.4.1",
"rxjs": "^5.5.2",
"zone.js": "^0.8.14"
},
"devDependencies": {
"@angular/cli": "1.6.3",
"@angular/compiler-cli": "^5.0.0",
"@angular/language-service": "^5.0.0",
"@types/jasmine": "~2.5.53",
"@types/jasminewd2": "~2.0.2",
"@types/node": "~6.0.60",
"codemlyzer": "^4.0.1",
"jasmine-core": "~2.6.2",
"jasmine-spec-reporter": "~4.1.0",
"karma": "~1.7.0",
"karma-chrome-launcher": "~2.1.1",
"karma-cli": "~1.0.1",
"karma-coverage-istanbul-reporter": "^1.2.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~3.2.0",
"tslint": "~5.7.0",
"typescript": "~2.4.2"
}
}
}

```

c:\angular\myapp\src\app\app.component.ts

```

import { Component } from "@angular/core";
import { trigger, state, style, transition, animate } from "@angular/animations";

var myanimations = [
trigger("animation1", [
state("visible", style({ transform: "scale(1)" })),
state("invisible", style({ transform: "scale(0)" })),
transition("visible => invisible", animate("500ms")),
transition("invisible => visible", animate("800ms")),
])
];
;

@Component({ selector: "app-root", templateUrl: "./app.component.html", animations: [myanimations] })
export class AppComponent
{
currentstate: string = "visible";

```

```
ShowData()
{
 this.currentState = "visible";
}

HideData()
{
 this.currentState = "invisible";
}
```

c:\angular\myapp\src\app\app.component.html

```
<div>
 <h1>Animations</h1>
 <input type="button" value="Show Data" (click)="ShowData()">
 <input type="button" value="Hide Data" (click)="HideData()">
 <div [@animation1]="currentState" style="background-color:#77cae8; height: 100px">
 Hello
 </div>
</div>
```

c:\angular\myapp\src\app\app.component.css

```
/* You can add css code of "app component" here */
```

c:\angular\myapp\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from "@angular/forms";
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

@NgModule({
 declarations: [AppComponent],
 imports: [BrowserModule, FormsModule, BrowserAnimationsModule],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule
{}
```

c:\angular\myapp\src\styles.css

```
/* You can add global styles to this file, and also import other style files */
```

```
c:\angular\myapp\src\main.ts
```

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production)
{
 enableProdMode();
}
platformBrowserDynamic().bootstrapModule(AppModule)
 .catch(err => console.log(err));
```

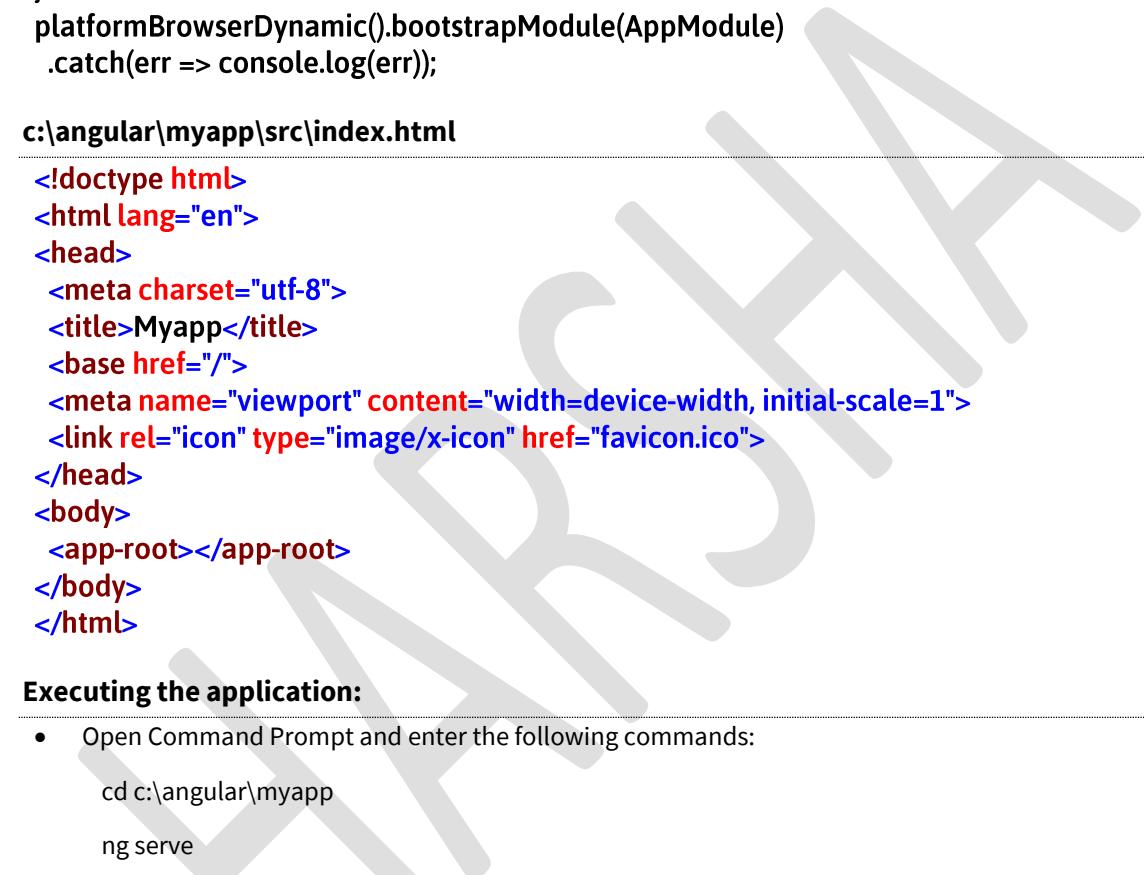
```
c:\angular\myapp\src\index.html
```

```
<!doctype html>
<html lang="en">
<head>
 <meta charset="utf-8">
 <title>Myapp</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></app-root>
</body>
</html>
```

### Executing the application:

- Open Command Prompt and enter the following commands:

```
cd c:\angular\myapp
ng serve
```

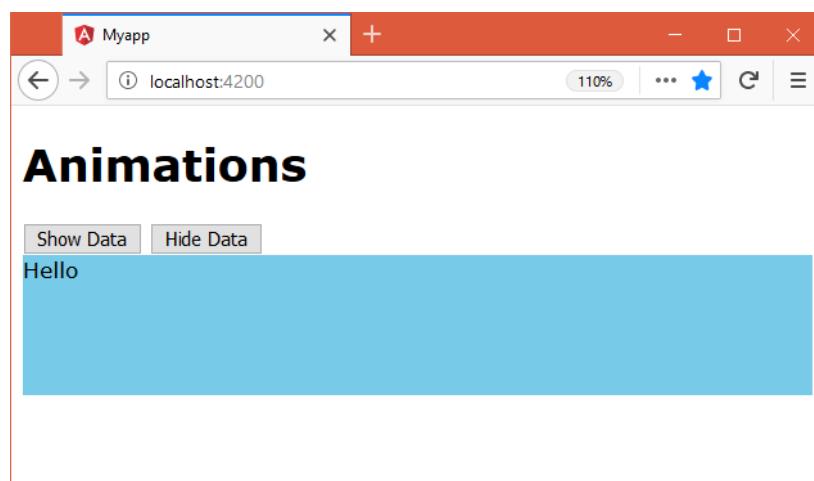


```
PS C:\Users\Harsha>cd c:\angular\myapp
C:\angular\myapp>ng serve
** NG Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
Date: 2018-01-10T14:37:36.618Z
Hash: f18ec8883de54f119b6f
Time: 8259ms
chunk {inline} inline.bundle.js (inline) 5.79 kB [entry] [rendered]
chunk {main} main.bundle.js (main) 16.9 kB [initial] [rendered]
chunk {polyfills} polyfills.bundle.js (polyfills) 553 kB [initial] [rendered]
chunk {styles} styles.bundle.js (styles) 33.6 kB [initial] [rendered]
chunk {vendor} vendor.bundle.js (vendor) 7.95 MB [initial] [rendered]

webpack: Compiled successfully.
```

- Open the browser and enter the following URL:

<http://localhost:4200>



## Angular Material Design

- "Angular Material Design" provides a set of UI components such as buttons, textboxes, checkboxes, radio button, dropdownlists, datepicker, menus etc., with rich look and feel.
- It is an alternative to other UI frameworks such as bootstrap and dojo toolkit.
- It provides "Ripple Effect", which is already available in Google Chrome and Android.

### Packages of Angular Material Design

- Angular Material Design can be implemented using the following set of packages:
  1. @angular/material
  2. @angular/cdk
  3. @angular/animations
  4. hammerjs

#### 1. @angular/material

- This package provides a set of modules which contains angular material design components.
- **List of modules:**
  - MatButtonModule: Used to create buttons.
  - MatInputModule: Used to create textboxes.
  - MatCheckBoxModule: Used to create checkboxes.
  - MatRadioModule: Used to create radio buttons.
  - MatSelectModule: Used to create dropdownlists.
  - MatAutoCompleteModule: Used to create comboboxes.
  - MatDatepickerModule: Used to create date pickers.

- MatSlideToggleModule: Used to slide toggle buttons.
- MatMenuModule: Used to create menus.
- MatIconModule: Used to display icons in buttons / menus.
- MatTableModule: Used to create tables.
- MatTabsModule: Used to create tabs.
- MatTooltipModule: Used to display tooltip messages.
- MatDialogModule: Used to dialog boxes / popup boxes.

### 2. @angular/cdk

- This package provides necessary API (Application Programming Interface), to create UI components, based on which the material design components are developed.

### 3. @angular/animations

- This package provides animations for angular material components.

### 4. hammerjs

- This package provides touch events for angular material components.

### Themes of Angular Material Design

- Angular Material Design provides the following set of pre-defined themes (css files), which contains styles of material components:
  1. indigo-pink.css
  2. pink-bluegrey.css
  3. purple-green.css
  4. deeppurple-amber.css

### Steps for setting-up Angular Material Design:

- Create a new application using Angular CLI

ng new app1

- Import necessary packages in “package.json” file:

```
"@angular/material": "^5.0.0",
"@angular/cdk": "^5.0.0",
"@angular/animations": "^5.0.0",
```

"hammerjs": "latest"

- Import icons in "src\index.html"

```
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
```

- Import theme (css file) in "src\styles.css"

```
@import "~@angular/material/prebuilt-themes/indigo-pink.css";
```

- Import "hammerjs" in "src\main.ts"

```
import "hammerjs";
```

- Import "BrowserAnimationsModule" in "src\app\app.module.ts"

```
import { BrowserAnimationsModule } from "@angular/platform-browser/animations";
```

```
@NgModule({
 declarations: [AppComponent],
 imports: [BrowserModule, BrowserAnimationsModule],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule
{}
```

## MatButtonModule

- The "MatButtonModule" is used to create buttons.

Syntax:     <button mat-button>text here</button>

- **Types of Buttons**

- mat-button : Basic Button
- mat-raised-button : Raised Button
- mat-fab : Fab Button
- mat-mini-fab : Mini Fab Button

- **Colors of Buttons**

- color="primary" : Fuchsia Blue Color
- color="accent" : Wild Strawberry Color

- color="warn" : Outrageous Orange Color

### MatButtonModule - Example

c:\angular\package.json

```
{
 "name": "materialapp",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 "build": "ng build --prod",
 "test": "ng test",
 "lint": "ng lint",
 "e2e": "ng e2e"
 },
 "private": true,
 "dependencies": {
 "@angular/common": "^5.0.0",
 "@angular/compiler": "^5.0.0",
 "@angular/core": "^5.0.0",
 "@angular/forms": "^5.0.0",
 "@angular/http": "^5.0.0",
 "@angular/platform-browser": "^5.0.0",
 "@angular/platform-browser-dynamic": "^5.0.0",
 "@angular/router": "^5.0.0",
 "core-js": "^2.4.1",
 "rxjs": "^5.5.2",
 "zone.js": "^0.8.14",
 "@angular/material": "^5.0.0",
 "@angular/cdk": "^5.0.0",
 "@angular/animations": "^5.0.0",
 "hammerjs": "latest"
 },
 "devDependencies": {
 "@angular/cli": "1.6.1",
 "@angular/compiler-cli": "^5.0.0",
 "@angular/language-service": "^5.0.0",
 "@types/jasmine": "~2.5.3",
 "@types/jasminewd2": "~2.0.2",
 "@types/node": "~6.0.60",
 "codelyzer": "^4.0.1",
 "jasmine-core": "~2.6.2",
 "jasmine-spec-reporter": "~4.1.0",
 "karma": "~1.7.0",
 "karma-chrome-launcher": "~2.1.1",
 "karma-cli": "~1.0.1",
 "karma-coverage-istanbul-reporter": "^1.2.1",
 "karma-jasmine": "~1.1.0",
 }
}
```

```
"karma-jasmine-html-reporter": "^0.2.2",
"protractor": "~5.1.2",
"ts-node": "~3.2.0",
"tslint": "~5.7.0",
"typescript": "~2.4.2"
}
}
```

c:\angular\src\index.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Materialapp</title>
<base href="/">
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
<app-root></app-root>
</body>
</html>
```

c:\angular\src\main.ts

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import "hammerjs";

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
 enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
 .catch(err => console.log(err));
```

c:\angular\src\app\app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { BrowserAnimationsModule } from "@angular/platform-browser/animations";
import { MatButtonModule, MatIconModule } from "@angular/material";
import { AppComponent } from './app.component';
import { ReactiveFormsModule } from "@angular/forms";

@NgModule({
 declarations: [
 AppComponent
],
 imports: [
```

```
imports: [
 BrowserModule, BrowserAnimationsModule, MatButtonModule, MatIconModule
],
providers: [],
bootstrap: [AppComponent]
})
export class AppModule {}

c:\angular\scripts\app.ts


```
import { Component, NgModule } from "@angular/core";
import { BrowserModule } from "@angular/platform-browser";
import { platformBrowserDynamic } from "@angular/platform-browser-dynamic";
import { FormsModule } from "@angular/forms";
import { trigger, state, style, transition, animate } from "@angular/animations";
import { BrowserAnimationsModule } from "@angular/platform-browser/animations";  
  
//animations
var myanimations = [
  trigger("animation1", [
    state("visible", style({ transform: "scale(1)" })),
    state("invisible", style({ transform: "scale(0)" })),
    transition("* => *", animate("500ms"))
  ])
];
  
//create parent component
@Component({ selector: "app", templateUrl: "AppComponentTemplate.html", animations: myanimations })
class AppComponent
{
  empid : number = 101;
  empname : string = "Scott";
  salary : number = 4500;
  currentstate : string = "visible";
  
  ShowData()
  {
    this.currentstate = "visible";
  }
  
  HideData()
  {
    this.currentstate = "invisible";
  }
}
//create module
@NgModule({ declarations: [ AppComponent ], imports: [ BrowserModule, FormsModule,
  BrowserAnimationsModule ], bootstrap: [ AppComponent ] })
class AppModule
{
}
//bootstrap module
platformBrowserDynamic().bootstrapModule(AppModule);
```


```

### c:\angular\AppComponentTemplate.html

```
<div>
 <h1>Animations</h1>
 <input type="button" value="Show Data" (click)="ShowData()">
 <input type="button" value="Hide Data" (click)="HideData()">

 <div [@animation1]="currentstate" style="background-color:#33ff66">
 Emp ID: {{empid}}

 Emp Name: {{empname}}

 Salary: {{salary}}
 </div>
</div>
```

### Executing the application:

- Open Command Prompt and enter the following commands:

```
cd c:\angular
npm install
tsc
http-server -c-0
```



A screenshot of a Windows Command Prompt window titled "Command Prompt - http-server -c-0". The window shows the following command-line session:

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Harsha>cd c:\angular
c:\angular>npm install
added 11 packages in 22.939s
c:\angular>tsc
c:\angular>http-server -c-0
Starting up http-server, serving .
Available on:
 http://192.168.0.5:8080
 http://127.0.0.1:8080
Hit CTRL-C to stop the server
>
```

- Open the browser and enter the following URL:

<http://localhost:8080>