Angular JS

Dr. Arul Xavier V.M

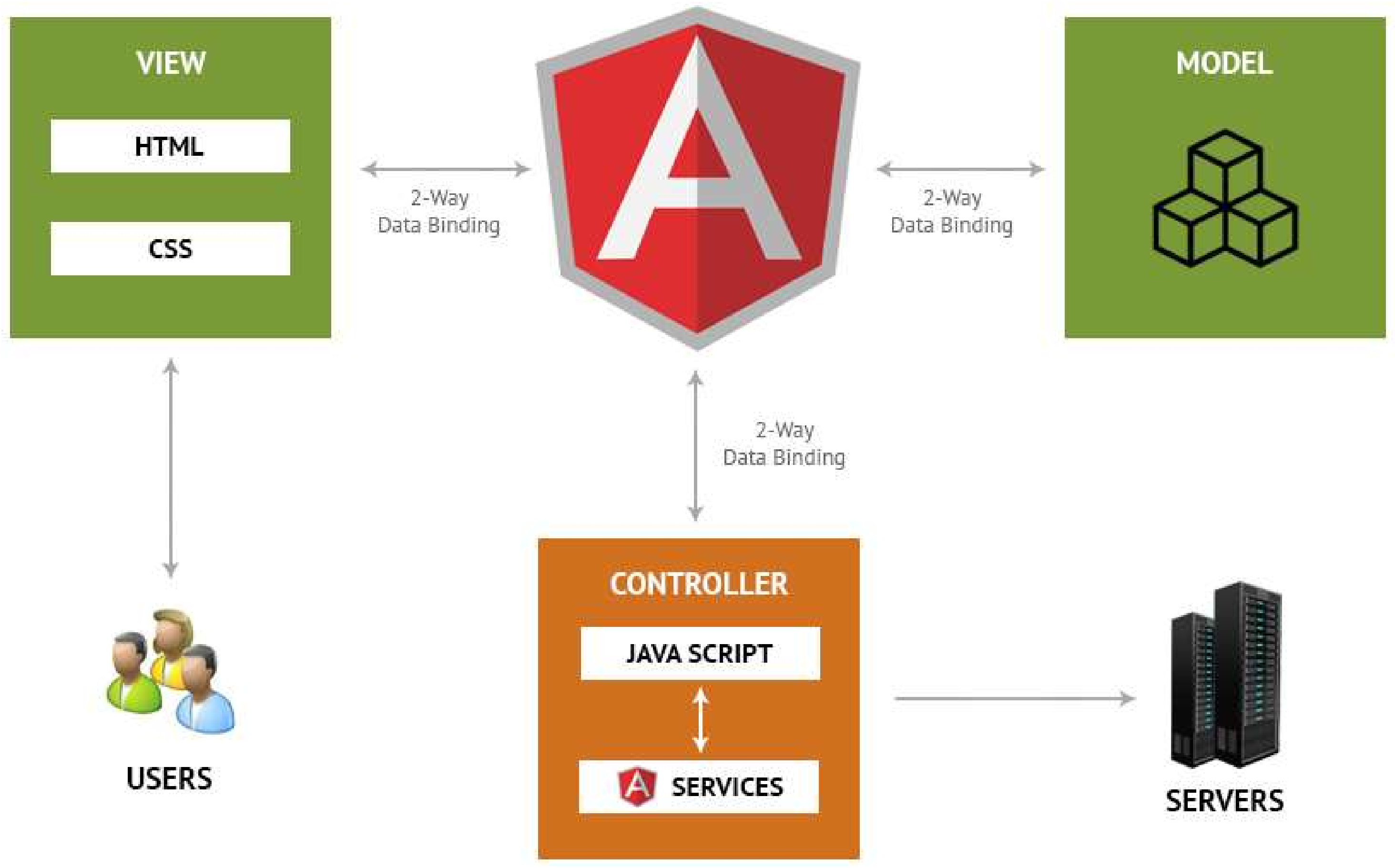
Introducing AngularJS

- AngularJS is a superheroic JavaScript Model View Controller(MVC) framework for the Web Application Developments.
- It is based on pure Javascript and HTML.
- AngularJS was created in 2009 by two developers, Misko Hevery and Adam Abrons.

MVC (Model-View-Controller)

- The core concept behind the AngularJS framework is the MVC architectural pattern.
- MVC stands for Model-View-Controller evolved as a way to separate data, logical units and presentation in web application development.
  - The *model* is the data behind the application, usually fetched from the server.
  - The *view* is the UI that the user sees and interacts with. It is dynamic, and generated based on the current model of the application.
  - The *controller* is the business logic and presentation layer, which performs actions such as fetching data, and makes decisions such as how to present the model, which parts of it to display, etc.

Angular JS – Model View Controller



AngularJS Benefits

- AngularJS is a Single Page Application (SPA) Framework.
- An AngularJS application will require fewer lines of code to complete a task than a pure JavaScript.
- AngularJS's declarative nature makes it easier to write and understand applications.
- AngularJS applications can be styled using CSS and HTML independent of their business logic and functionality.
- AngularJS application templates are written in pure HTML

Integration of Angular JS

- It can be added to an HTML page with a <script> tag.
- Angular JS provides set of Directives as HTML attributes.
- Angular JS provides Expressions to bind the data in HTML view page.

Starting Out with AngularJS

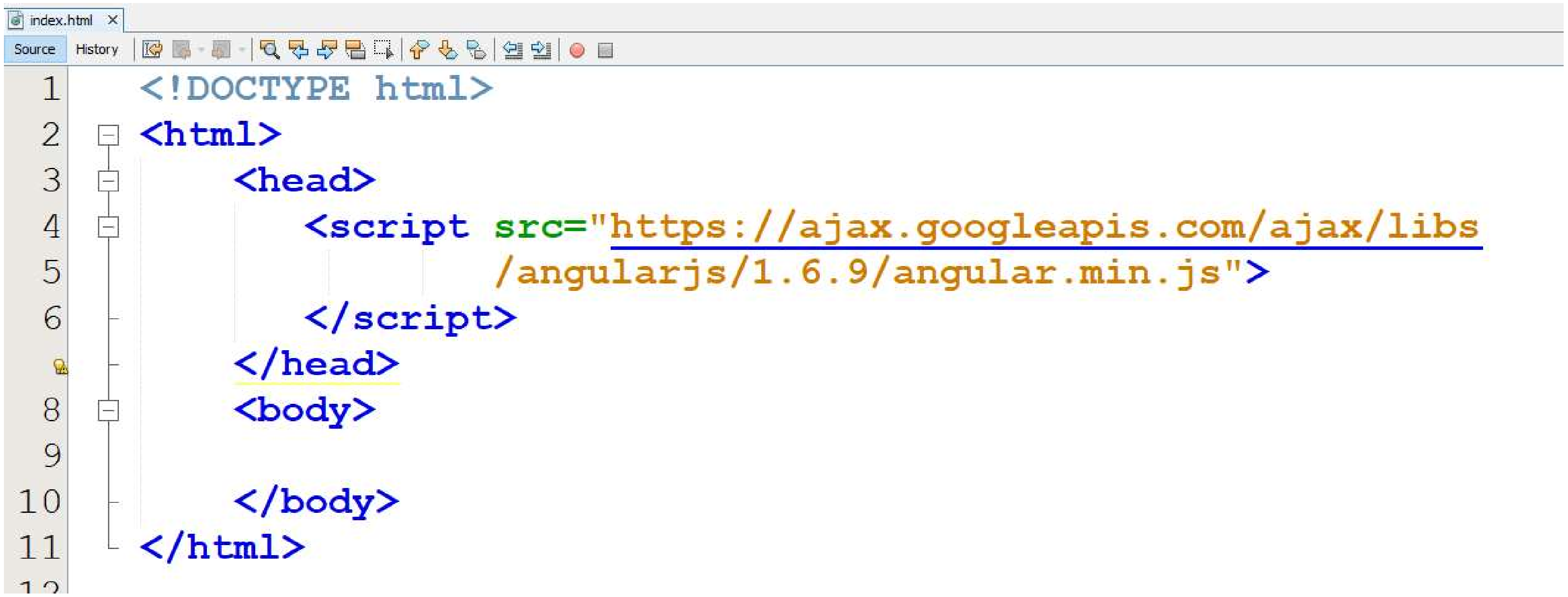
- AngularJS Extends HTML
  - AngularJS extends HTML with ng-directives.
  - The ng-app
    - directive defines an AngularJS application
  - The ng-init
    - directive used to create initial value(model) for the angular JS application.
  - The ng-model
    - directive binds the value of HTML controls (input, select, textarea) to application data.
  - The ng-bind
    - directive binds application data to the HTML view.

Integrating Angular JS in HTML

- The angular JS can be included via script tag which just imports the AngularJS library and proves that AngularJS is bootstrapped and working:

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js">

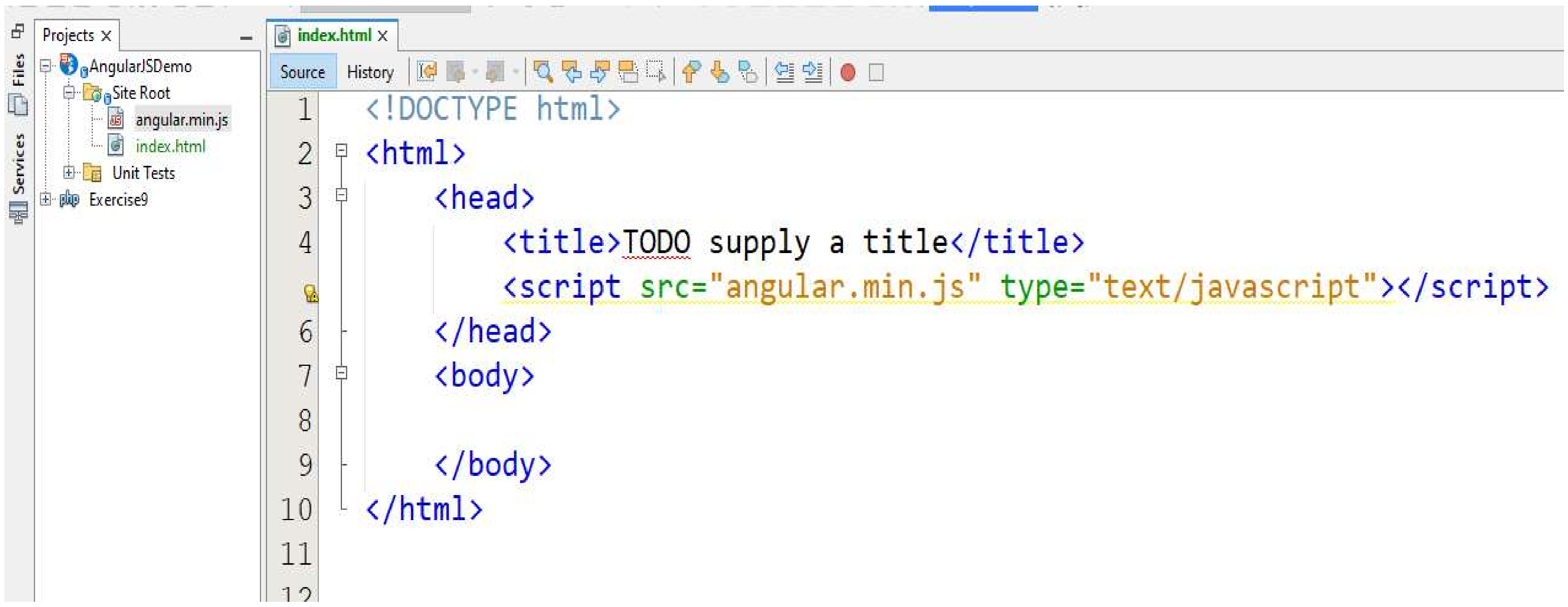
</script>



```
<!DOCTYPE html>
<html>
    <head>
        <script src="https://ajax.googleapis.com/ajax/libs
                /angularjs/1.6.9/angular.min.js">
        </script>
    </head>
    <body>

    </body>
</html>
```

Another way to add Angular JS

- Download the angular.min.js file and include using <script> tag.



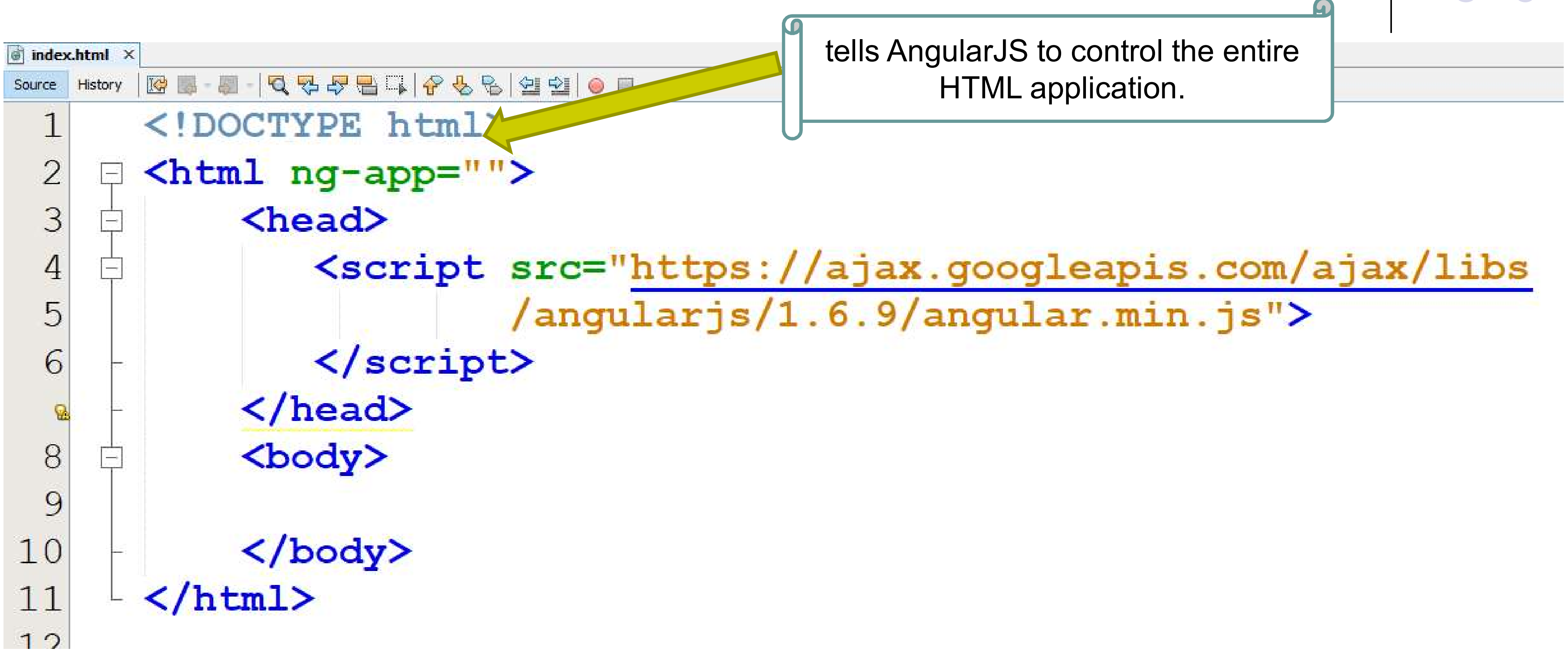
```
<!DOCTYPE html>
<html>
    <head>
        <title>TODO supply a title</title>
        <script src="angular.min.js" type="text/javascript"></script>
    </head>
    <body>

    </body>
</html>
```

Getting Started with AngularJS

- This is done through the ng-app directive.
  - This is the first and most important directive that AngularJS has, which denotes the section of HTML that AngularJS controls.
  - Putting it on the **<html>** tag tells AngularJS to control the entire HTML application.
  - We could also put it on the **<body>** or any other element on the page such as **<div>, <p>** and etc.
  - Any element that is a child of that will be also handled with AngularJS and **anything outside would not be processed**.

Getting Started with AngularJS

ng-app

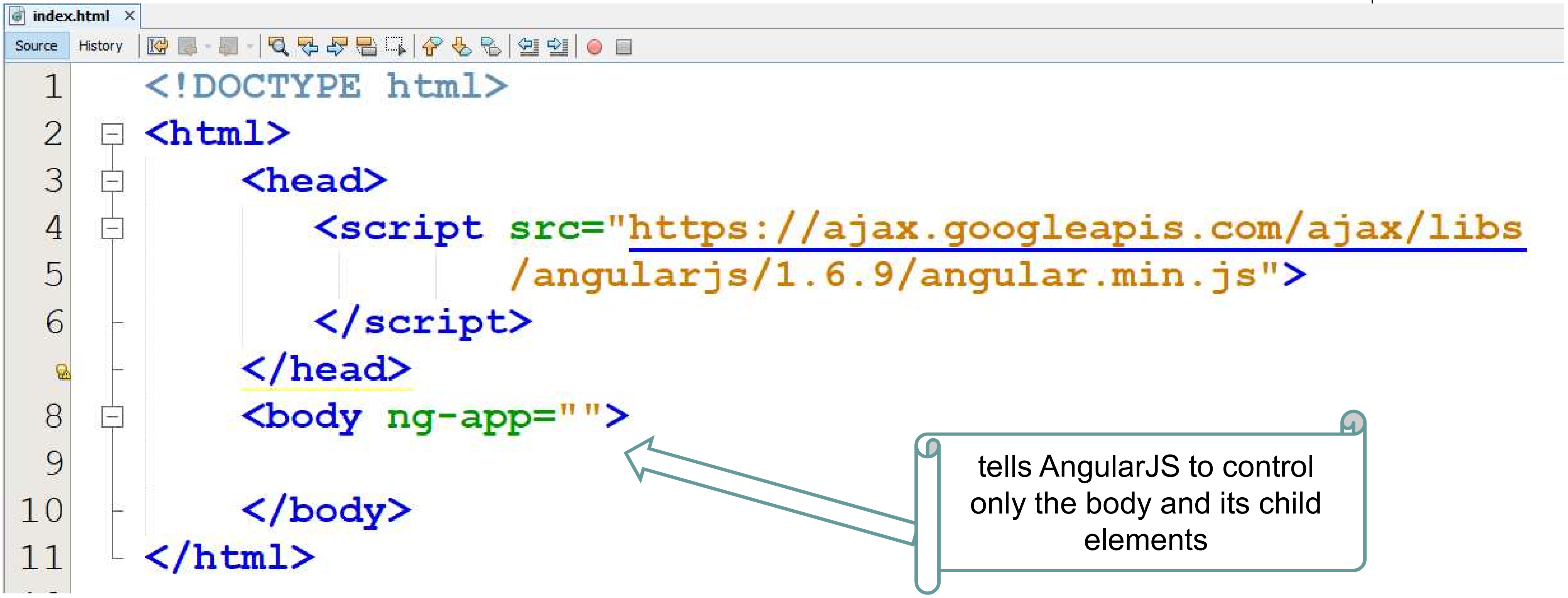


```
<!DOCTYPE html>
<html ng-app="">
    <head>
        <script src="https://ajax.googleapis.com/ajax/libs
                /angularjs/1.6.9/angular.min.js">
        </script>
    </head>
    <body>

    </body>
</html>
```

tells AngularJS to control the entire HTML application.

Getting Started with AngularJS

ng-app

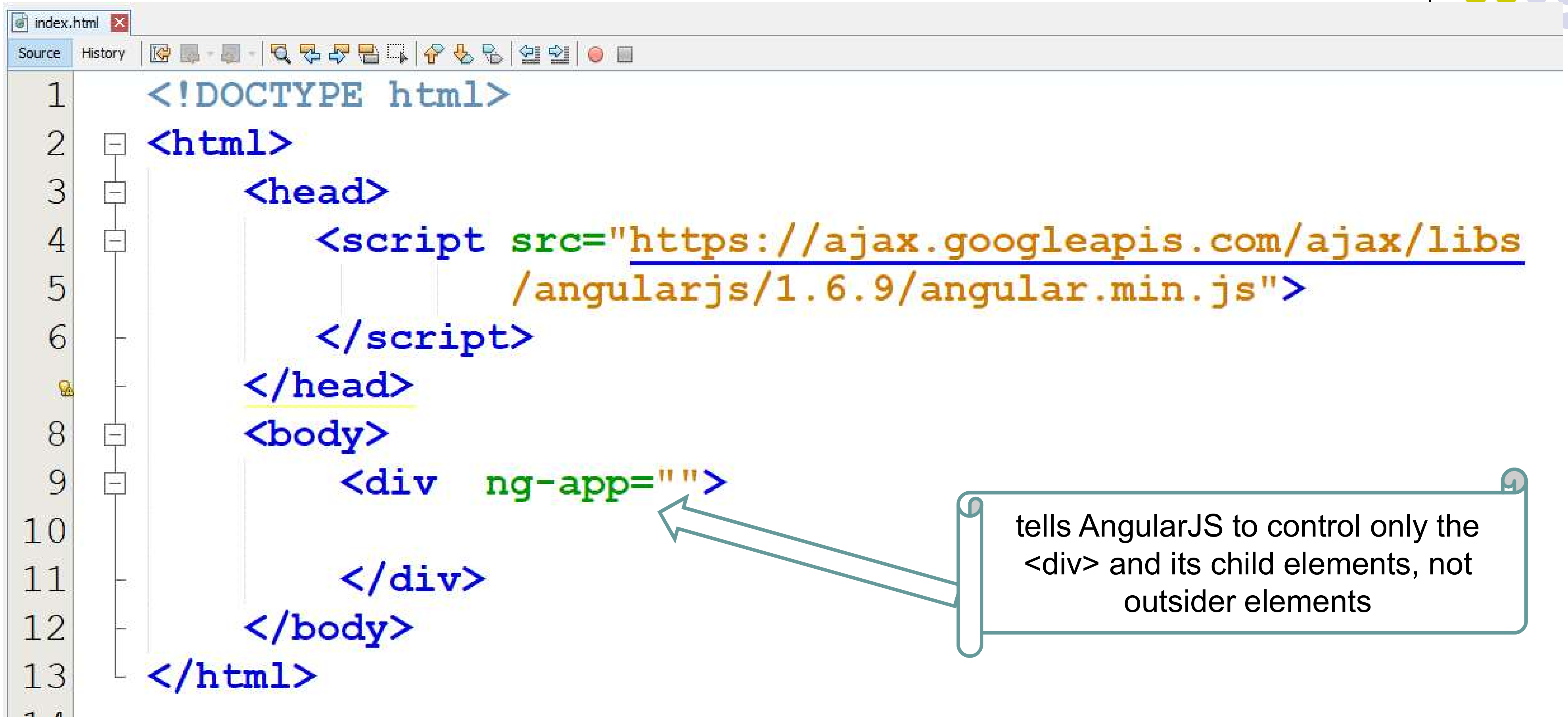


```
<!DOCTYPE html>
<html>
    <head>
        <script src="https://ajax.googleapis.com/ajax/libs
                /angularjs/1.6.9/angular.min.js">
        </script>
    </head>
    <body ng-app="">

    </body>
</html>
```

tells AngularJS to control only the body and its child elements

Getting Started with AngularJS

ng-app



index.html

Source History

```
<!DOCTYPE html>
<html>
    <head>
        <script src="https://ajax.googleapis.com/ajax/libs
                /angularjs/1.6.9/angular.min.js">
        </script>
    </head>
    <body>
        <div  ng-app="">

        </div>
    </body>
</html>
```

tells AngularJS to control only the <div> and its child elements, not outsider elements

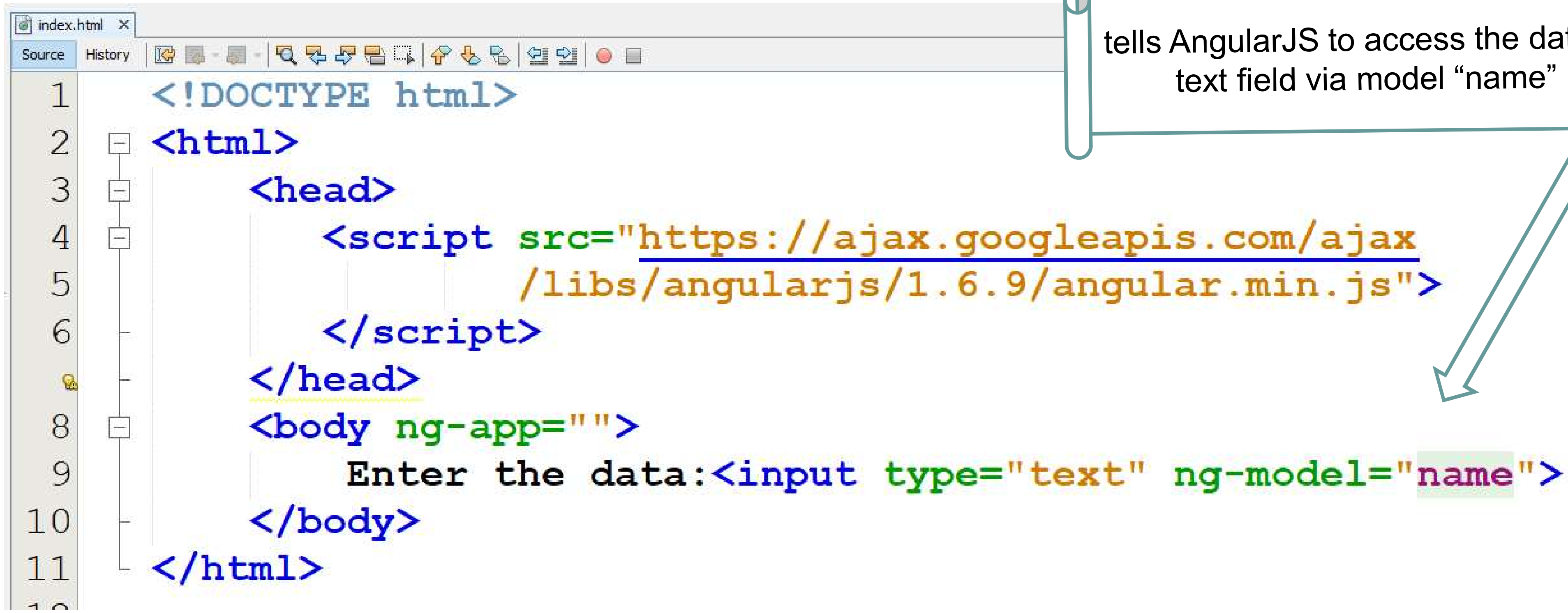
AngularJS ng-init Directive

- We can create initial data model such values, arrays when initiating the application.

```
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-app="" ng-init="data=100">

</body>
</html>
```

Creating data model

- The ng-model
  - directive binds the value of HTML controls (input, select, textarea) to application data.



```
<!DOCTYPE html>
<html>
    <head>
        <script src="https://ajax.googleapis.com/ajax
                /libs/angularjs/1.6.9/angular.min.js">
        </script>
    </head>
    <body ng-app="">
        Enter the data:<input type="text" ng-model="name">
    </body>
</html>
```

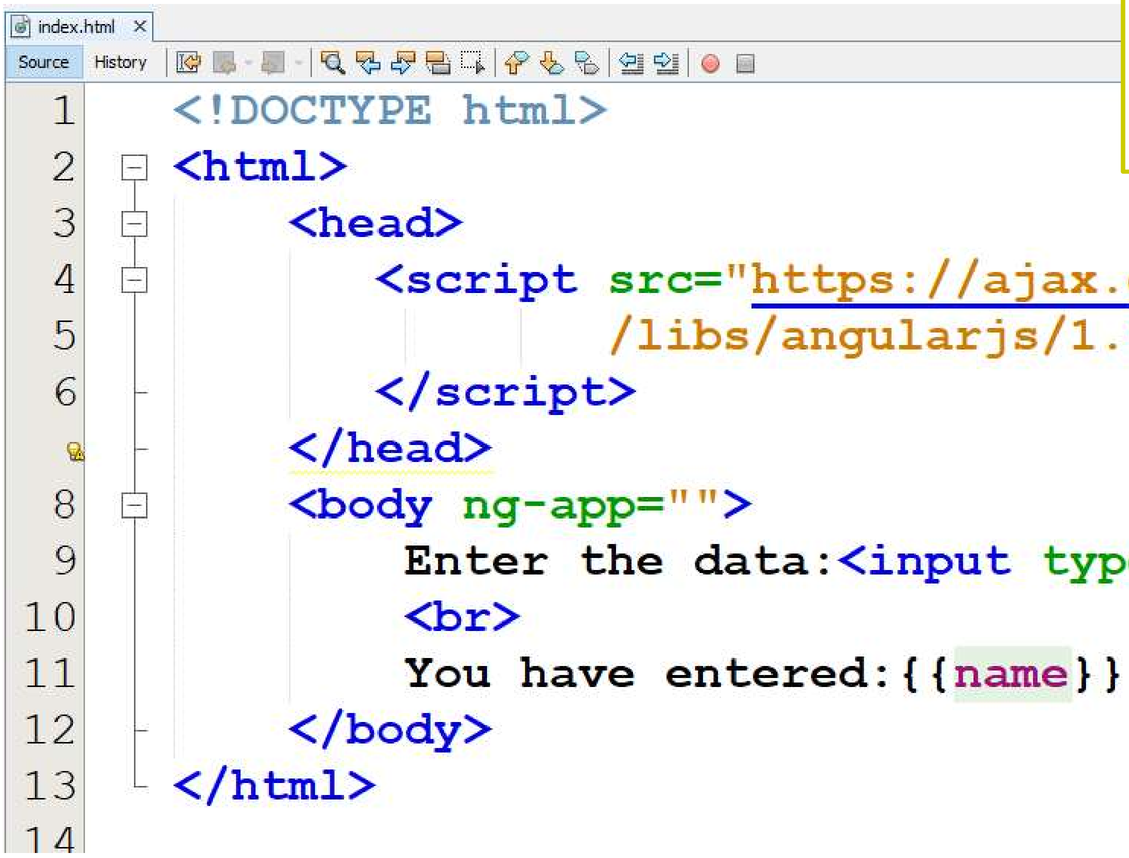
tells AngularJS to access the data of text field via model “name”

Binding data model to HTML view

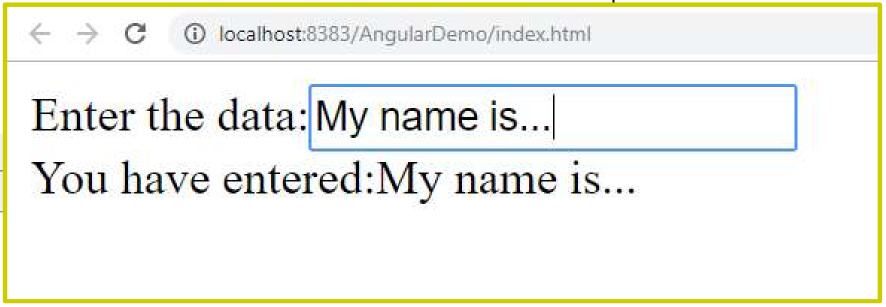
- Binding data to HTML view can be done in two ways
  - Using double curly expression
    - {{model_name}}
  - Using ng-bind directive
    - <p ng-bind=“model_name”></p>

Binding data model to HTML view

- Using double curly expression



```
<!DOCTYPE html>
<html>
    <head>
        <script src="https://ajax.googleapis.com/ajax
                /libs/angularjs/1.6.9/angular.min.js">
        </script>
    </head>
    <body ng-app="">
        Enter the data:<input type="text" ng-model="name">
        <br>
        You have entered:{{name}}
    </body>
</html>
```

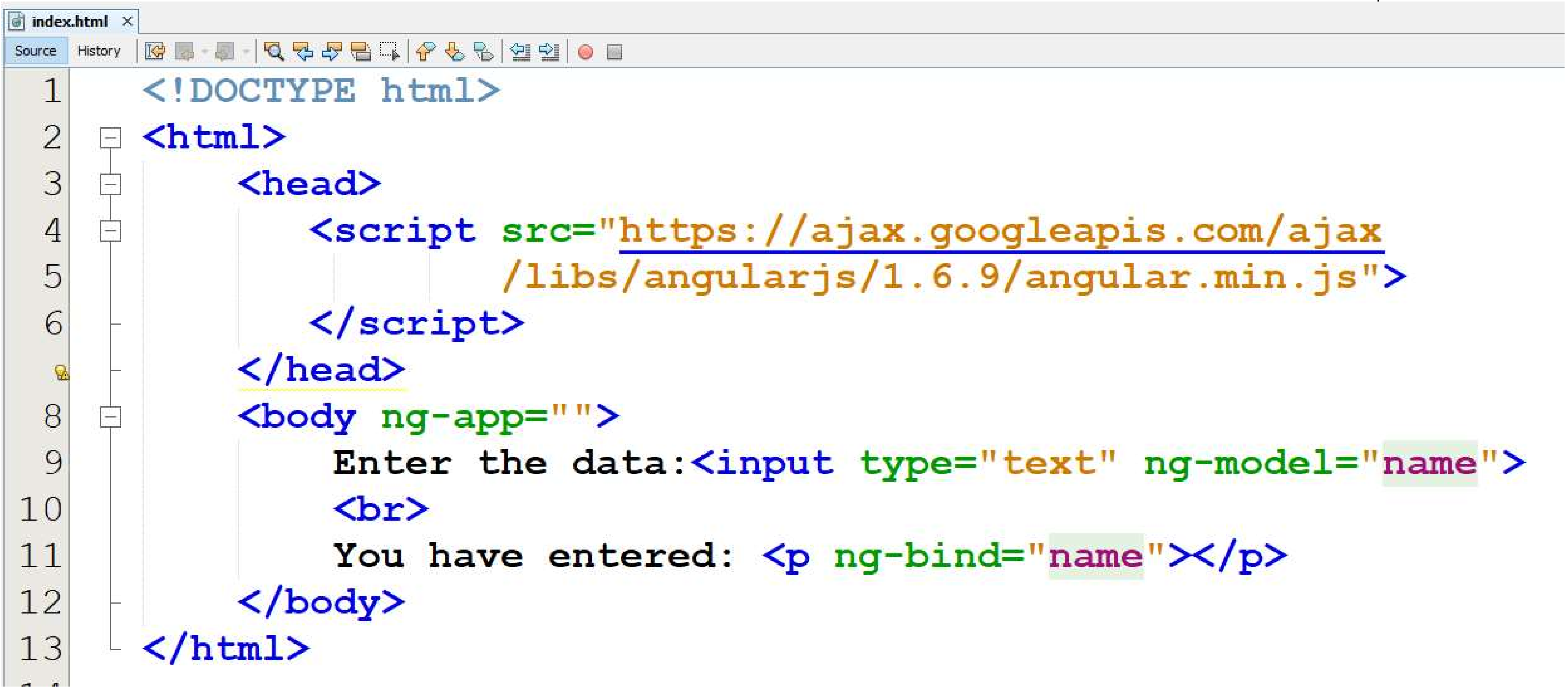


localhost:8383/AngularDemo/index.html

```
Enter the data: My name is...
You have entered:My name is...
```

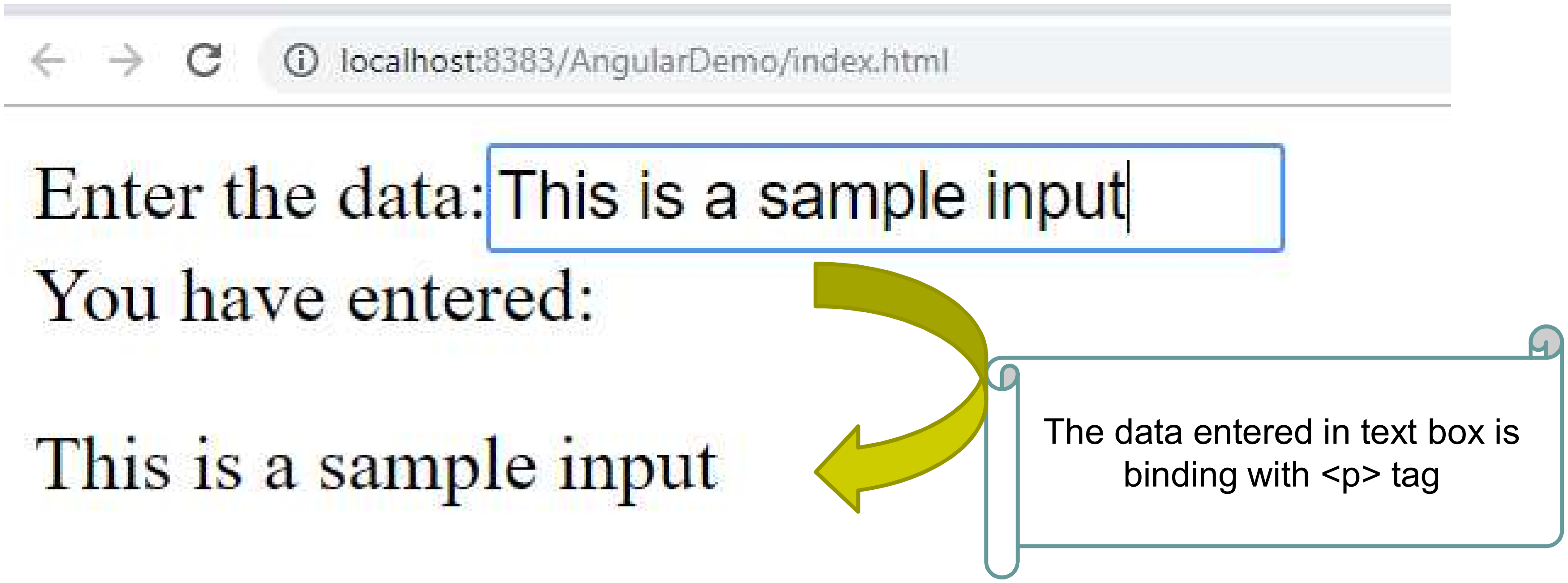
Binding data model to HTML view

- Using ng-bind directive



```
<!DOCTYPE html>
<html>
    <head>
        <script src="https://ajax.googleapis.com/ajax
                /libs/angularjs/1.6.9/angular.min.js">
        </script>
    </head>
    <body ng-app="">
        Enter the data:<input type="text" ng-model="name">
        <br>
        You have entered: <p ng-bind="name"></p>
    </body>
</html>
```

Binding data model to HTML view - Sample Output



localhost:8383/AngularDemo/index.html

Enter the data: This is a sample input

You have entered:

This is a sample input

The data entered in text box is binding with <p> tag

AngularJS Modules

- Modules are AngularJS's way of packaging relevant code under a single name.
- An AngularJS module defines an application.
- The module is a container for the different parts of an application.
- The module is a container for the application Controllers.
  - Controllers always belong to a module.

AngularJS Modules

- In addition to being a container for related JavaScript, the module is also what AngularJS uses to bootstrap an application.
  - What that means is that we can tell AngularJS what module to load as the main entry point for the application by passing the module name to the ng-app directive.
  - The ng-app directive takes an optional argument, which is the name of the module to load during bootstrapping.

Creating an Angular JS module

- A module is created by using the AngularJS function angular.module

```
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
    <script>
        var app = angular.module("myapp",[]);
    </script>
</head>
<body ng-app="myapp">

</body>
</html>
```

It creates the module, first argument "name of the module" and second argument is an array of additional libraries.

Tells angular JS to control <body> contents using the newly created module

Creating First Controller

- Controllers in AngularJS used to create business logic, the JavaScript functions that perform or trigger the majority of our UI-oriented work.

Creating First Controller

```
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="angular.min.js"></script>
    <script>
        var app = angular.module("myapp",[]);
        app.controller('mycontroller',function(){
            //logic goes here
        })
    </script>
</head>
<body ng-app="myapp" ng-controller="mycontroller">

</body>
</html>
```

Creating data in Controller

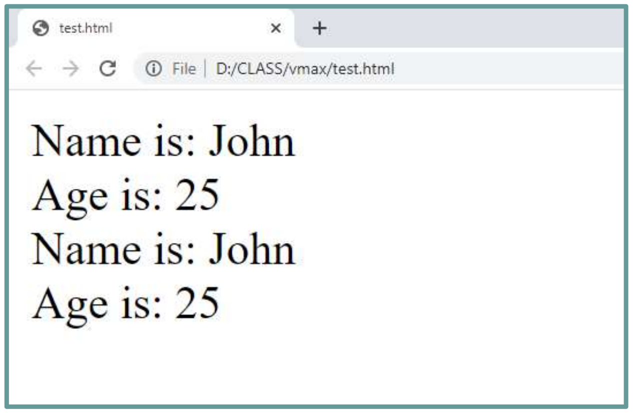
- $scope object can be used to create data inside controller.
- Later, this data can be binded in HTML view elements.

```
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="angular.min.js"></script>
    <script>
         var app = angular.module("myapp",[]);
         app.controller('mycontroller',function($scope){
              $scope.name = "John";
              $scope.age = 25;
         })
    </script>
</head>
<body ng-app="myapp" ng-controller="mycontroller">
</body>
</html>
```

Binding data from controller to HTML view

- Once we create a controller variable, you can bind the data using the variable names.

```
<!DOCTYPE html>
<html>
<head>
    <script src="angular.min.js"></script>
    <script>
        var app = angular.module("myapp",[]);
        app.controller('mycontroller',function($scope){
            $scope.name = "John";
            $scope.age = 25;
        })
    </script>
</head>
<body ng-app="myapp" ng-controller="mycontroller">
    <div>Name is: {{name}}</div>
    <div>Age is: {{age}}</div>
    <div>Name is: <span ng-bind="name"></span></div>
    <div>Age is: <span ng-bind="age"></span></div>
</body>
</html>
```

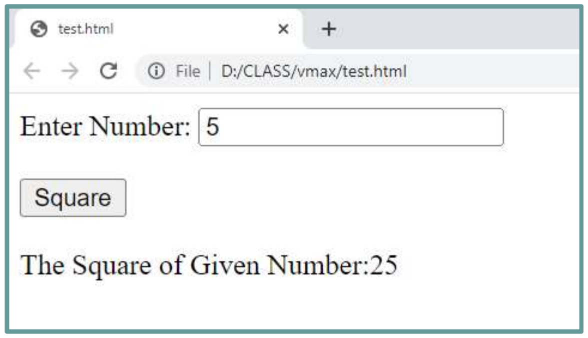


```
Name is: John
Age is: 25
Name is: John
Age is: 25
```

Adding methods inside Controller

- User defined function can be included in controller using $scope object.
- The user defined function can be called using ng-click directive

```
<html>
<head>
    <script src="angular.min.js"></script>
    <script>
         var app = angular.module("myapp",[]);
         app.controller('mycontroller',function($scope){
               $scope.findSquare = function(){
                  $scope.result = $scope.data ** 2;
               }
         })
    </script>
</head>
<body ng-app="myapp" ng-controller="mycontroller">
    Enter Number: <input type="text" ng-model="data"><br><br>
    <button ng-click="findSquare()">Square</button><br><br>
    <div>The Square of Given Number:{{result}}</div>
</body>
</html>
```



Enter Number: 5

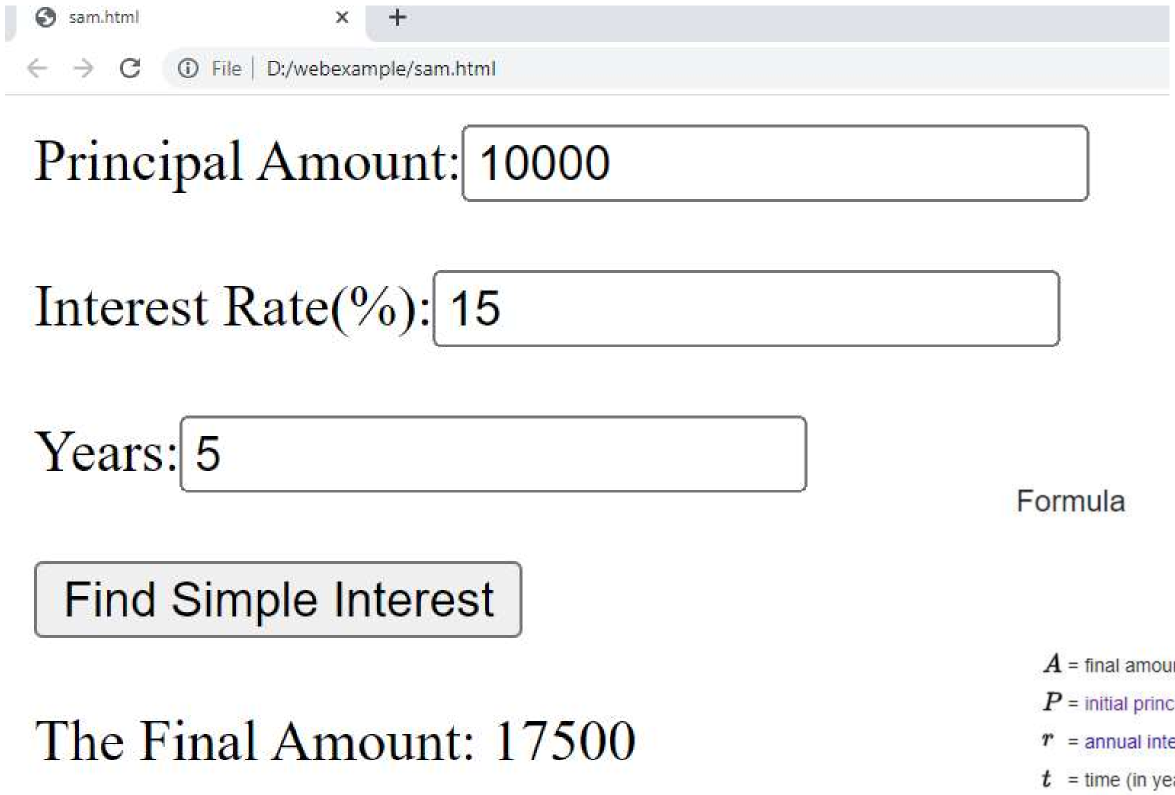
The Square of Given Number:25

ng-click directive example

- ng-click directive can be used to trigger function from HTML button element.

```
<script>
    var app = angular.module("myapp",[]);
    app.controller("mycontrol",function($scope){
        $scope.findSI = function(){
            $scope.output = $scope.p * (1+($scope.r/100)*$scope.t);
        }
    })
</script>

<body ng-app="myapp" ng-controller="mycontrol">
    Principal Amount:<input type="text" ng-model="p"><br><br>
    Interest Rate(%):<input type="text" ng-model="r"><br><br>
    Years:<input type="text" ng-model="t"><br><br>
    <button ng-click="findSI()">Find Simple Interest</button><br><br>
    The Final Amount: <span ng-bind="output"></span>
</body>
```

Simple Interest Calculator



Principal Amount: 10000

Interest Rate(%): 15

Years: 5

The Final Amount: 17500

Formula

$$A = P(1 + rt)$$

A = final amount

P = initial principal balance

r = annual interest rate

t = time (in years)

Working with Arrays and Displaying Arrays

- We have seen how to create a controller, and how to get data from the controller into the HTML.
- But we worked with very simplistic string messages. Let’s now take a look at how we would work with a collection of data;
- Collection of data is represented as arrays in angular JS.
- Array elements can be represented using square brackets [ ]

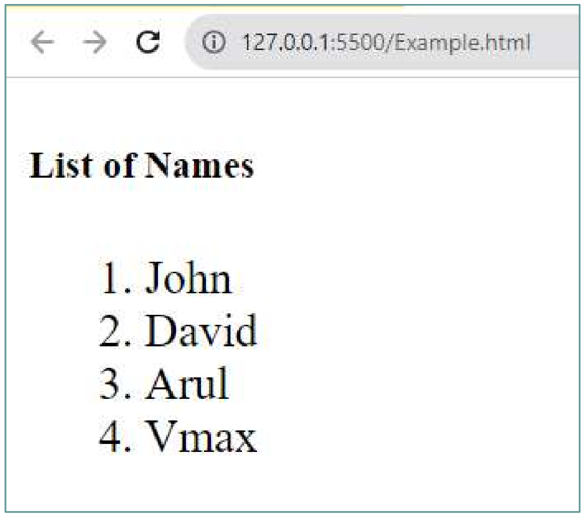
Creating Arrays in Angular JS

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="angular.min.js"></script>
    <script>
        var app = angular.module("MyApp",[])
        app.controller("MyController",function($scope){
            //Empty Array
            $scope.items = []

            //Array with Numbers
            $scope.numbers = [10,20,40,59,79]

            //Array with Text Data
            $scope.names = ['John','David','Arul','Vmax']
        })
    </script>
</head>
<body ng-app="MyApp" ng-controller="MyController"> </body>
</html>
```

Binding Array Elements to HTML View

- ng-repeat directives used to iterate over an array and display them in the HTML view. ng-repeat="eachVar in arrayVar"

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="angular.min.js"></script>
    <script>
        var app = angular.module("MyApp",[])
        app.controller("MyController",function($scope){
            $scope.names = ['John','David','Arul','Vmax']
        })
    </script>
</head>
<body ng-app="MyApp" ng-controller="MyController">
    <h5>List of Names</h5>
    <ol>
        <li ng-repeat="item in names">
            {{item}}
        </li>
    </ol>
</body>
</html>
```


127.0.0.1:5500/Example.html

List of Names

1. John
2. David
3. Arul
4. Vmax

ng-repeat with track by ID

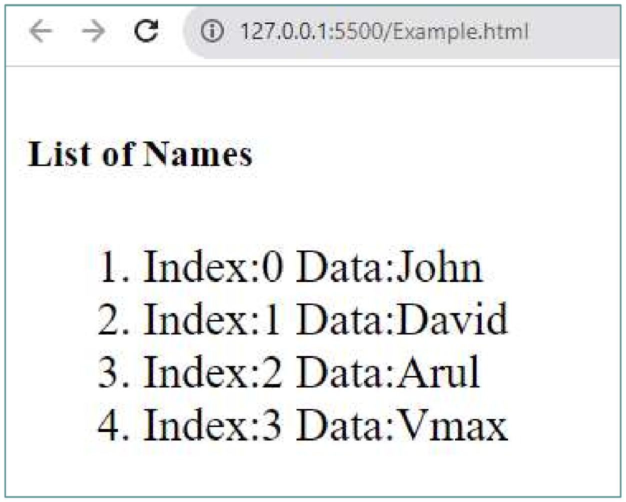
- track by $index can be used to retrieve array elements with $index when array contains duplicates items.

```
<script>
    var app = angular.module("myapp",[]);
    app.controller("mycontrol",function($scope){
       $scope.data = [1,2,2,3,4,4,5];
    })
</script>

<body ng-app="myapp" ng-controller="mycontrol">
    <p ng-repeat="x in data track by $index">
        {{"Data: " + x + " Index:" + $index }}
    </p>
</body>
```

$index in ng-repeat

- The ng-repeat directive also exposes some helper variables which allows us to gain some insight into the current element.
  - $index -> gives us the index or position of the item in the array.

```
<script>
    var app = angular.module("MyApp",[])
    app.controller("MyController",function($scope){
        $scope.names = ['John','David','Arul','Vmax']
    })
</script>
<ol>
     <li ng-repeat="item in names">
         Index:{{$index}}
         Data:{{item}}
     </li>
 </ol>
```



List of Names

```
1. Index:0 Data:John
2. Index:1 Data:David
3. Index:2 Data:Arul
4. Index:3 Data:Vmax
```

Angular JS Array Methods

- push(data) – used for inserting data in an array
  - Example:-

```
$scope.items = []
 //adding single data
 $scope.items.push(10)
 //adding multiple data using javascript object
 $scope.items.push({name:'Iphone',price:78000})
```

- splice(index,count) – used for deleting data in an array

```
$scope.items = [10,45,67,89,89]

//to delete 67
$scope.items.splice(2,1)
```

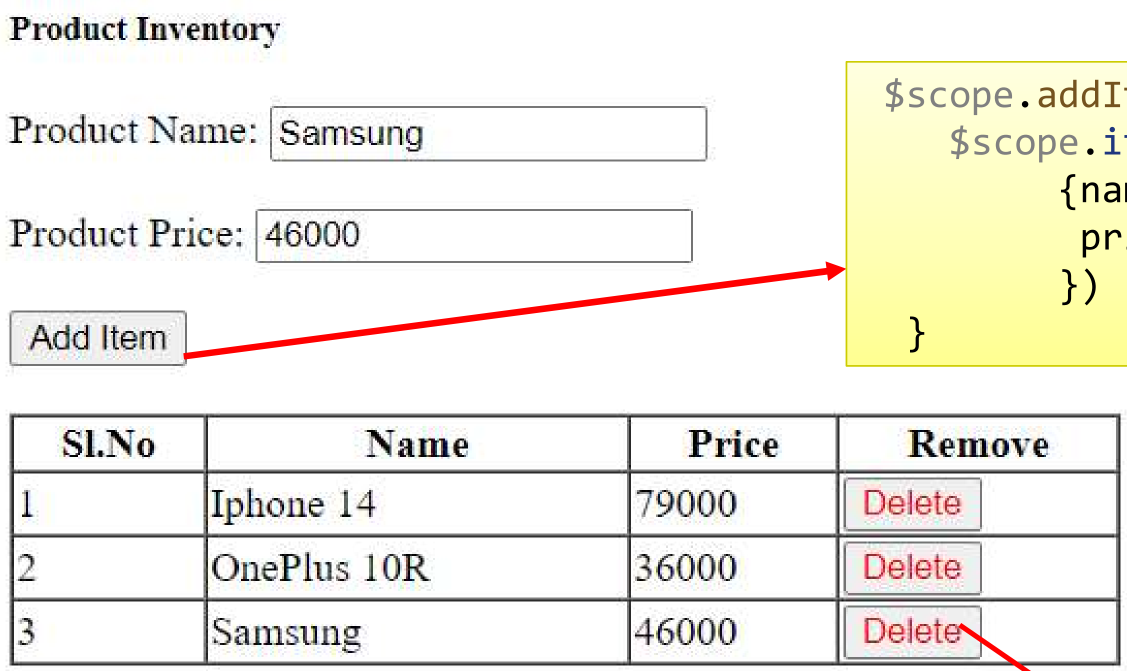
Creating a Product Inventory using ng-controller and array

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="angular.min.js"></script>
    <script>
        var app = angular.module("MyApp",[])
        app.controller("MyController",function($scope){
            $scope.items = []
            $scope.addItem = function(){
                $scope.items.push({name:$scope.name,price:$scope.price})
            }
            $scope.removeItem = function(index){
                $scope.items.splice(index,1)
            }
        })
    </script>
</head>
```

Creating a Shopping Cart using ng-controller and array

```
<body ng-app="MyApp" ng-controller="MyController">
     <h5>Product Inventory</h5>
     Product Name: <input type="text" ng-model="name"><br><br>
     Product Price: <input type="text" ng-model="price"><br><br>
     <button ng-click="addItem()">Add Item</button><br><br>
     <table border="1" cellspacing="0" width="40%">
        <tr><th>Sl.No</th><th>Name</th><th>Price</th><th>Remove</th></tr>
        <tr ng-repeat="data in items">
            <td>{{$index+1}}</td>
            <td>{{data.name}}</td>
            <td>{{data.price}}</td>
            <td>
                <button ng-click="removeItem($index)" style="color:red">
                   Delete
                </button>
            </td>
        </tr>
     </table>
</body>
```

Output:



Product Inventory

Product Name: Samsung

Product Price: 46000

Add Item

Sl.No	Name	Price	Remove
1	Iphone 14	79000	Delete
2	OnePlus 10R	36000	Delete
3	Samsung	46000	Delete

```
$scope.addItem = function(){
   $scope.items.push(
        {name:$scope.name,
         price:$scope.price
        })
 }
```

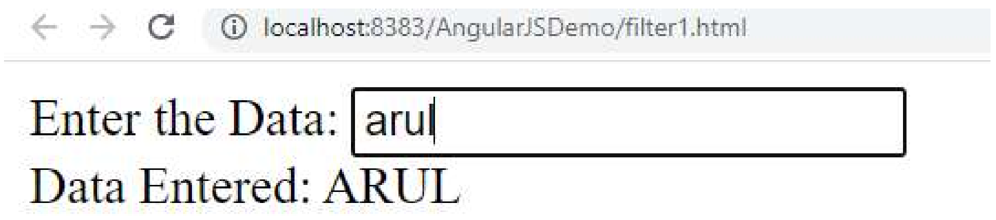
```
  $scope.removeItem = function(index){
                $scope.items.splice(index,1)
          }
```

Angular JS Filters

- Filters can be added in AngularJS to format data.
- Example filters are:-
  - lowercase - Format a string to lower case.
  - uppercase - Format a string to upper case.
  - orderBy - Orders an array by an expression.
  - date - Format a date to a specified format.
  - currency – Format as $ by default.

uppercase / lowercase filter

```
<body>
    <div ng-app="">
        Enter the Data: <input type="text" ng-model="data"><br>
        <div>Data Entered: {{data|uppercase}} </div>
    </div>
</body>
```


localhost:8383/AngularJSDemo/filter1.html

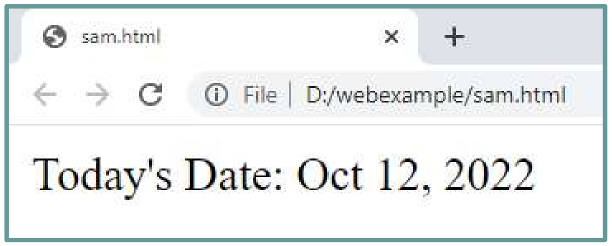
Enter the Data: arul

Data Entered: ARUL

date filter

- The date filter formats a date to a specified format.

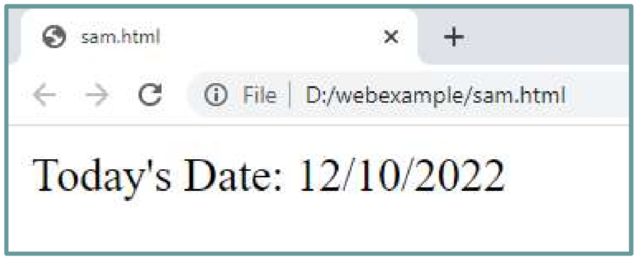
```
<script>
    var app = angular.module("myapp",[]);
    app.controller("mycontrol",function($scope){
         $scope.value = new Date();
    })
</script>
<body ng-app="myapp" ng-controller="mycontrol">
     <div>Today's Date: <span ng-bind="value | date "></span></div>
</body>
```


sam.html

File | D:/webexample/sam.html

```
Today's Date: Oct 12, 2022
```

```
<body ng-app="myapp" ng-controller="mycontrol">
     <div>Today's Date: <span ng-bind="value | date:'dd/MM/yyyy'"></span></div>
</body>
```

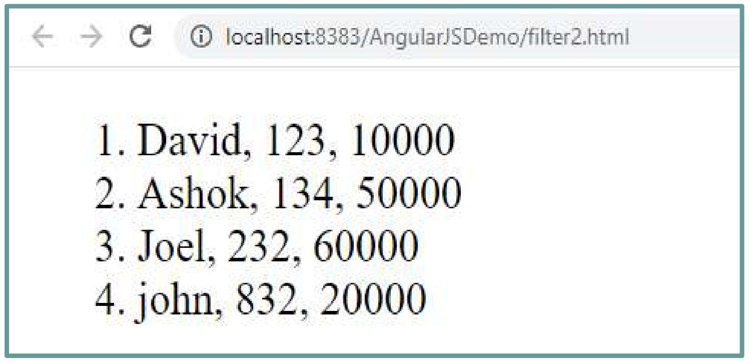

sam.html

File | D:/webexample/sam.html

```
Today's Date: 12/10/2022
```

orderBy filter

```
 <script>
    var app = angular.module('myapp',[]);
    app.controller('mycontrol',function($scope){
          $scope.employees = [
              {name:'john',empid:832,salary:20000},
              {name:'Joel',empid:232,salary:60000},
              {name:'David',empid:123,salary:10000},
              {name:'Ashok',empid:134,salary:50000}
          ];
    });
</script>
<body>
    <div ng-app="myapp" ng-controller="mycontrol">
        <ol>
            <li ng-repeat="item in employees | orderBy:'empid'">
                {{item.name}}, {{item.empid}}, {{item.salary}}
            </li>
        </ol>
    </div>
</body>
```

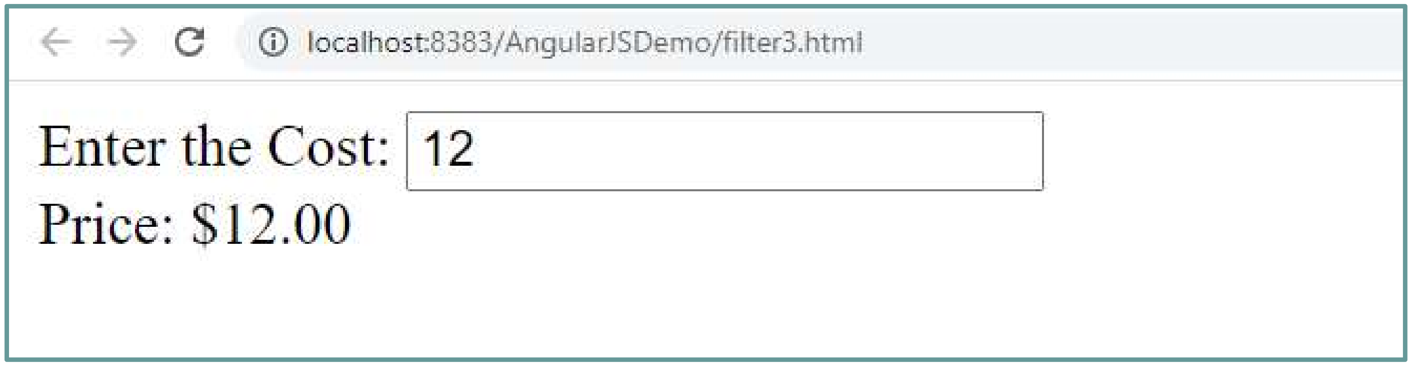


```
1. David, 123, 10000
2. Ashok, 134, 50000
3. Joel, 232, 60000
4. john, 832, 20000
```

currency filter

- To represent as currency format, By default, the locale currency format is used.

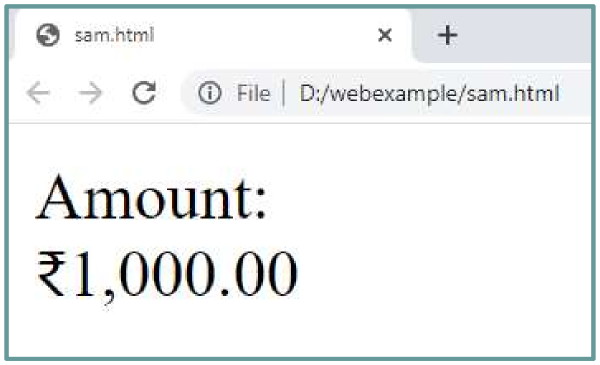
```
<div ng-app="">
    Enter the Cost: <input type="text" ng-model="data"><br>
    <div>Price: {{data|currency}} </div>
</div>
```



```
Enter the Cost: 12
Price: $12.00
```

To change the currency format

```
<script>
    var app = angular.module("myapp",[]);
    app.controller("mycontrol",function($scope){
        $scope.amount = 1000;
    })
</script>

<body ng-app="myapp" ng-controller="mycontrol">
    Amount: <div ng-bind="amount|currency:'₹'"></div>
</body>
```



```
Amount:
₹1,000.00
```

Form Validation and States

- AngularJS offers client-side form validation.
- AngularJS monitors the state of the form and input fields (input, textarea, select), and lets you notify the user about the current state.
- AngularJS also holds information about whether they have been touched, or modified, or not.
- You can use standard HTML5 attributes to validate input.

HTML Form Validators

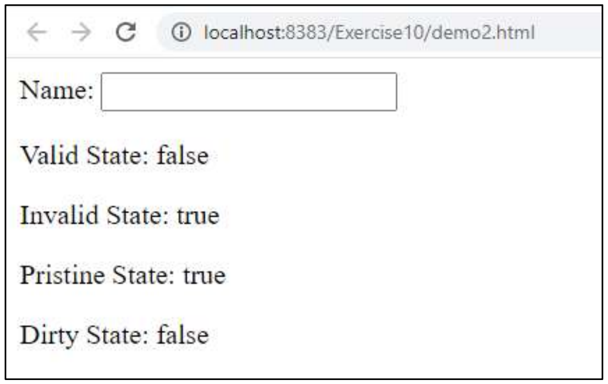
- type="email“
  - Text input with built-in email validation.
- required
  - Ensures that the field is required, and the field is marked invalid until it is filled out.

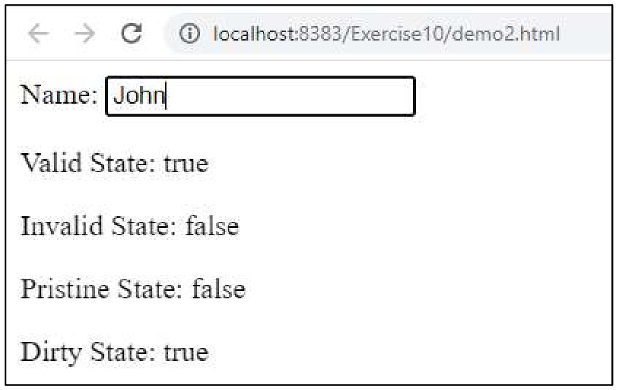
Validation States

- AngularJS is constantly updating the state of both the form and the input fields via FormController object.
- You can access the FormController for a form using the form's name.
- They are all properties of the input field, and are either true or false.
- Form States
  - $valid
    - The field content is valid
  - $invalid
    - The field content is not valid
  - $pristine
    - The field has not been modified yet.
  - $dirty
    - The field has been modified

Example

```
<html>
    <head>
        <script src="angular.min.js" type="text/javascript"></script>
    </head>
    <body ng-app="">
        <form name='myform'>
            Name: <input type='text' name="myName" ng-model="name" required>
            <p>Valid State: {{myform.myName.$valid}}</p>
            <p>Invalid State: {{myform.myName.$invalid}}</p>
            <p>Pristine State: {{myform.myName.$pristine}}</p>
            <p>Dirty State: {{myform.myName.$dirty}}</p>
        </form>
    </body>
</html>
```


```
localhost:8383/Exercise10/demo2.html

Name: 

Valid State: false

Invalid State: true

Pristine State: true

Dirty State: false
```


```
localhost:8383/Exercise10/demo2.html

Name: John

Valid State: true

Invalid State: false

Pristine State: false

Dirty State: true
```

Show Error/Success Message,
Disable the Button with respect to form states

```
<html>
    <head>
        <script src="angular.min.js" type="text/javascript"></script>
    </head>
    <body ng-app="">
        <form name='myform'>
            Name: <input type='text' name="myName" ng-model="name" required>
            <span style='color:red' ng-show='myform.myName.$dirty && myform.myName.$invalid'>
               Name is required..
            </span>
            <span style='color:green' ng-show='myform.myName.$dirty && myform.myName.$valid'>
              √
             </span>
            <br><br>
            <button ng-disabled="myform.myName.$invalid" ng-click="submit()">submit</button>
        </form>
    </body>
</html>
```

Displaying Error Messages

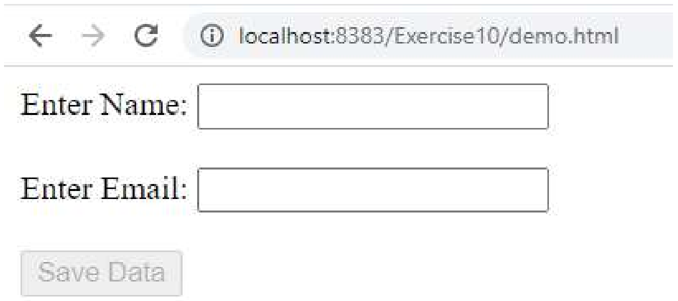
- The application needs to tell the user what went wrong and how to fix it.
- $error can be used to detect the correct error and show error message using ng-show directive.
- $error can be used with any HTML or Angular JS validators.
  - Example
    - $error.required
    - $error.email

Display Error Message

- **ng-show** directive is used to toggle between hide and show based on the validation states.
- It helps to show error message during validation.

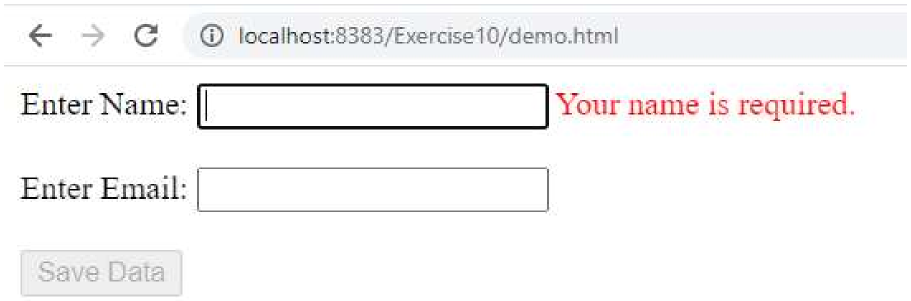
```
<html>
  <head>
     <script src="angular.min.js" type="text/javascript"></script>
  </head>
  <body ng-app="">
     <form name="myform">
        Enter Name: <input type="text" name='name' ng-model="username" required>
        <span style="color:red" ng-show="myform.name.$dirty && myform.name.$error.required">
           Your name is required.
        </span>
        <br><br>
        Enter Email: <input type="email" name='email' ng-model='useremail' required>
        <span style="color:red" ng-show="myform.email.$dirty && myform.email.$error.required">
            Your email is required.
        </span>
        <span style="color:red" ng-show="myform.email.$dirty && myform.email.$error.email">
            Email address wrong..
        </span>
        <br><br>
        <button ng-disabled="myform.name.$invalid || myform.email.$invalid" ng-click='submit()'>
            Save Data
        </button>
    </form>
  </body>
</html>
```

Error


localhost:8383/Exercise10/demo.html

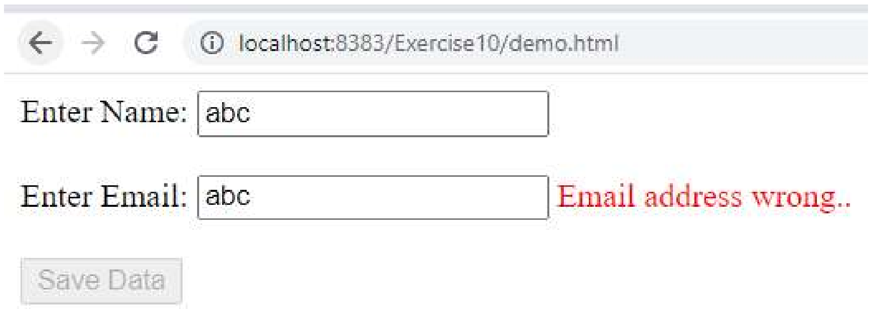
Enter Name:

Enter Email:


localhost:8383/Exercise10/demo.html

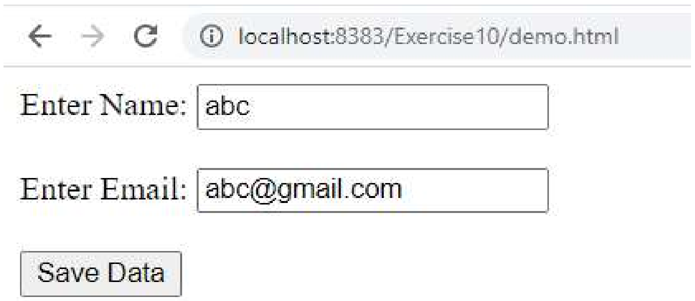
Enter Name: Your name is required.

Enter Email:


localhost:8383/Exercise10/demo.html

Enter Name: abc

Enter Email: abc Email address wrong..


localhost:8383/Exercise10/demo.html

Enter Name: abc

Enter Email: abc@gmail.com

Angular JS Routing

- The ngRoute module helps your application to become a Single Page Application.
  - If you want to navigate to different pages in your application, but you also want the application to be a SPA (Single Page Application), with no page reloading, you can use the ngRoute module.
  - The ngRoute module routes your application to different pages without reloading the entire application.

How to do Routing in Angular JS?

- To make your applications ready for routing, you must include the AngularJS Route module:

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular-route.js"></script>
```

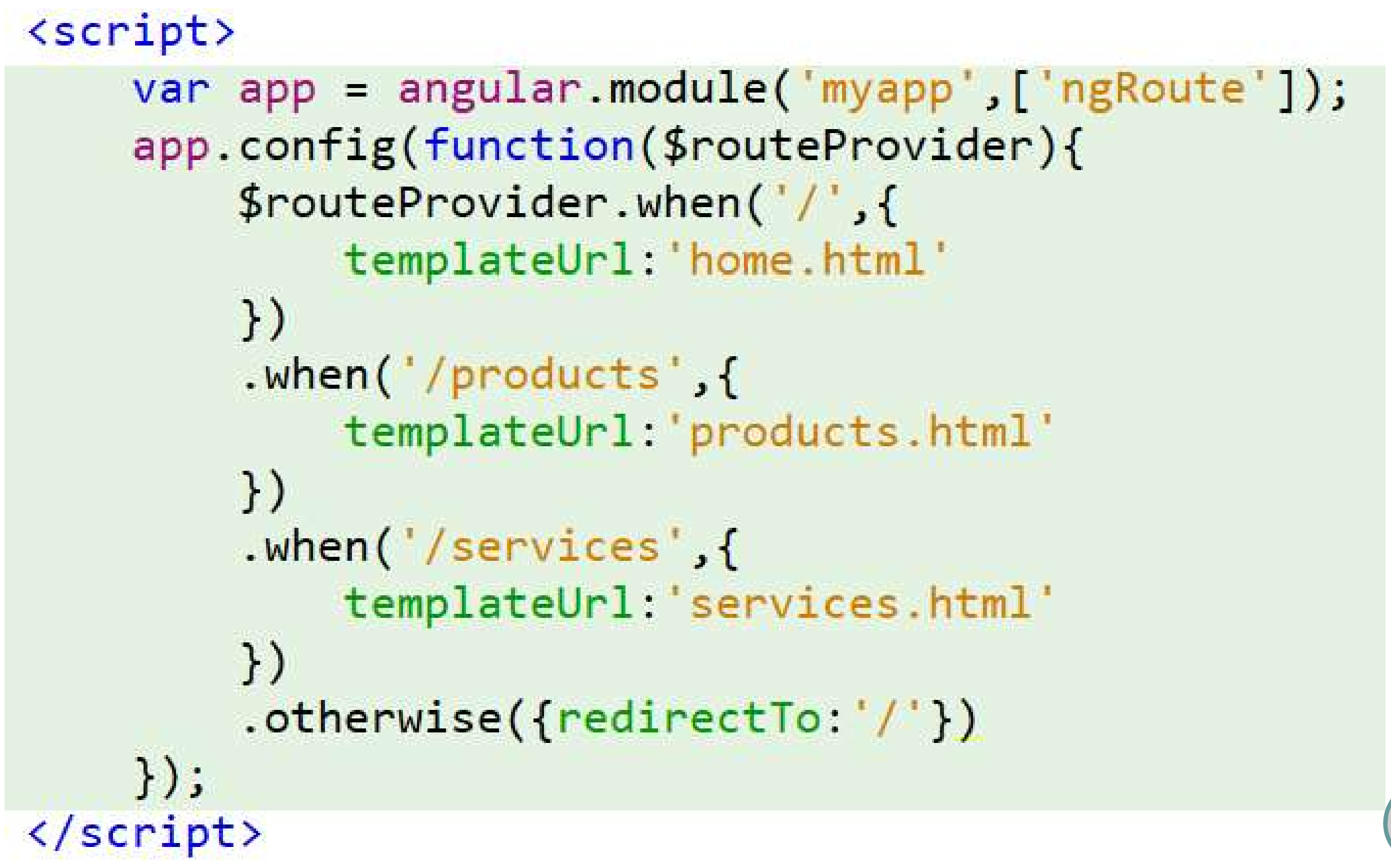
- Then you must add the ngRoute as a dependency in the application module:

```
var app = angular.module("myapp", ["ngRoute"]);
```

Implement Routing using $routeProvider

- Now your application has access to the route module, which provides the $routeProvider.
- Use the $routeProvider to configure different routes in your application by using config() method of Angular module.

Implement Routing using $routeProvider



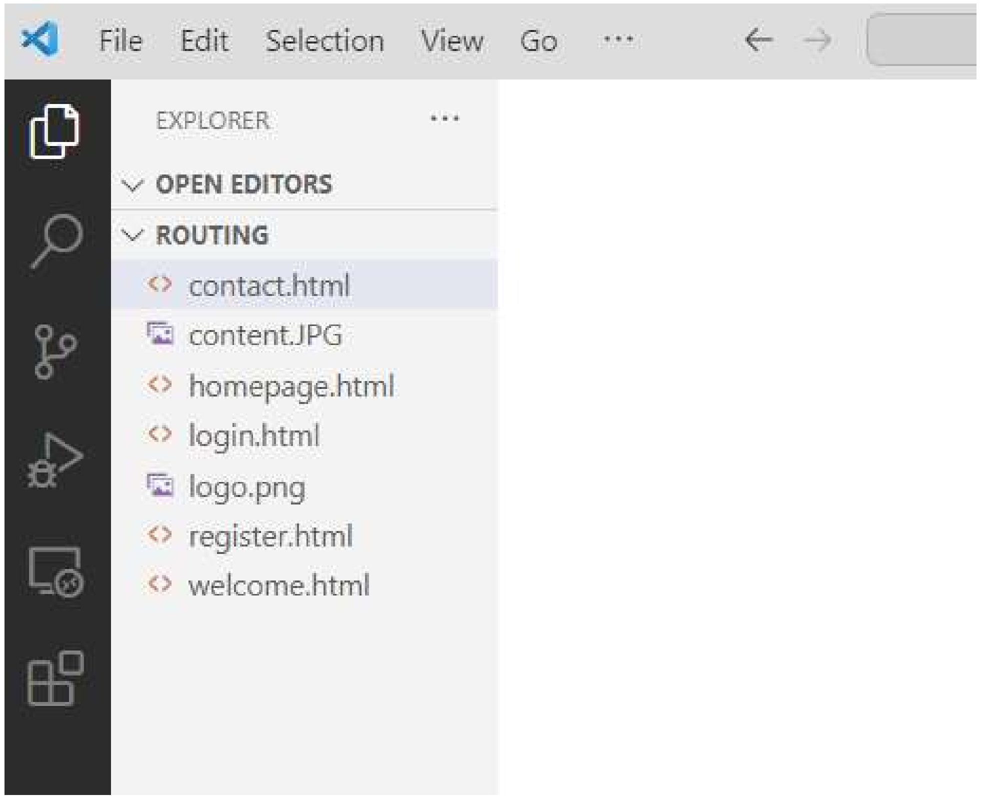
```
<script>
    var app = angular.module('myapp',['ngRoute']);
    app.config(function($routeProvider){
        $routeProvider.when('/',{
            templateUrl:'home.html'
        })
        .when('/products',{
            templateUrl:'products.html'
        })
        .when('/services',{
            templateUrl:'services.html'
        })
        .otherwise({redirectTo:'/'})
    });
</script>
```

1. When you navigate to ‘/’ it will take the content from ‘home.html’.
2. When you navigate to ‘/products’ it will take the content from ‘products.html’.
3. When you navigate to ‘/services’ it will take the content from ‘services.html’.

Updating View in Routing

- Your application needs a container to put the content provided by the routing.
- This container is the ng-view directive.
- There are three different ways to include the ng-view directive in your application:
  - `<div ng-view></div>`
  - `<ng-view></ng-view>`
  - `<div class="ng-view"></div>`

Example Program



- Create 4 html files
  - homepage.html - main page
  - welcome.html
  - login.html
  - register.html
  - contact.html

homepage.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Home Page</title>
     <script src="angular.min.js"></script>
     <script src="angular-route.js"></script>
     <script>
         var app = angular.module("MyApp",["ngRoute"])
         app.controller("MyController",function(){})
         app.config(function($routeProvider){
            $routeProvider.when("/",{
                templateUrl: 'welcome.html'
            })
            .when("/login",{
                templateUrl: 'login.html'
            })
            .when("/register",{
                templateUrl: 'register.html'
            })
            .when("/contact",{
                templateUrl: 'contact.html'
            })
         })
     </script>
</head>
```

“/” – root url

homepage.html

```
<body ng-app="MyApp" ng-controller="MyController">
    <h1 style="text-align: center;">Welcome to HOME Page</h1>
    <hr>
    <div>
        <a href="#!/">Welcome Page</a>
        <a href="#!login">Login</a>
        <a href="#!register">Register</a>
        <a href="#!contact">Contact Us</a>
    </div>
    <hr>
    <div ng-view>
       <!-- Content Goes Here... -->
    </div>
</body>
</html>
```

“#!/” – root url routing

welcome.html

```
<div>
    <img src="logo.png" alt="">
</div>
    <div style="height:300px;background-image: url('content.JPG');
               background-repeat: no-repeat;background-size: cover;">
</div>
```

login.html

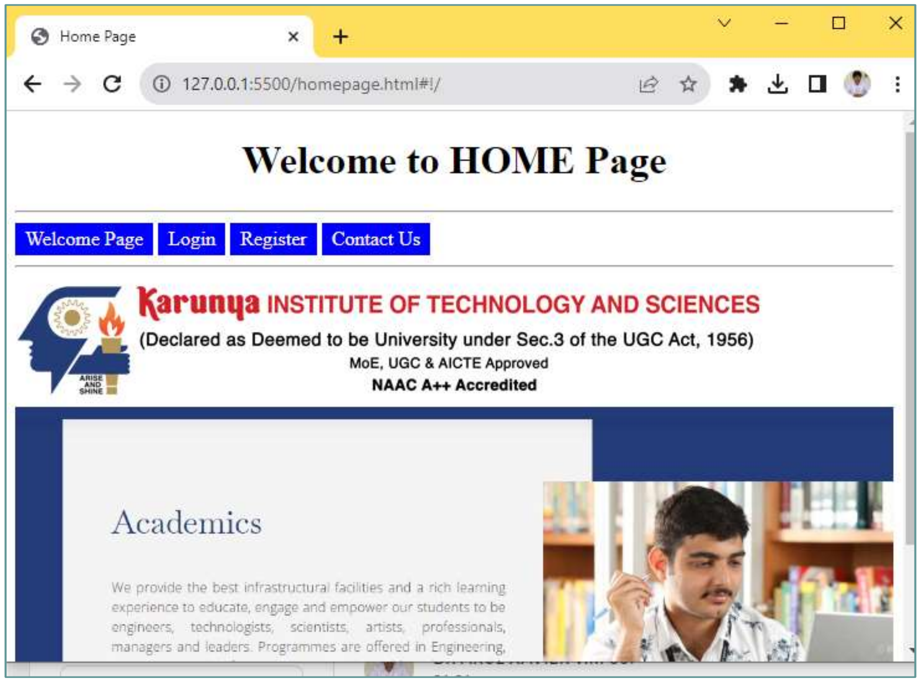
```
<div style="background-color: pink;padding: 20px;">
    <h4>Login Here</h4>
    <form>
        Username: <input type="text"> <br><br>
        Password: <input type="password"><br><br>
        <button>Login</button>
    </form>
</div>
```

register.html

```
<div style="background-color: gainsboro;padding: 20px;">
    <h4>New User Register Here</h4>
    <form>
        Name: <input type="text"> <br><br>
        Email: <input type="email"><br><br>
        Phone: <input type="text"><br><br>
        Select Password: <input type="password"><br><br>
        <button>Register</button>
    </form>
</div>
```

contact.html

```
<h5>Contact Details Page</h5>
<div style="margin: 20px;background-color: azure;">
    Karunya Institute of Technology and Sciences,<br>
    (Deemed to be University),<br>
    Karunya Nagar,<br>
    Coimbatore - 641 114,<br>
    Tamil Nadu, India<br>
</div>
```



Home Page

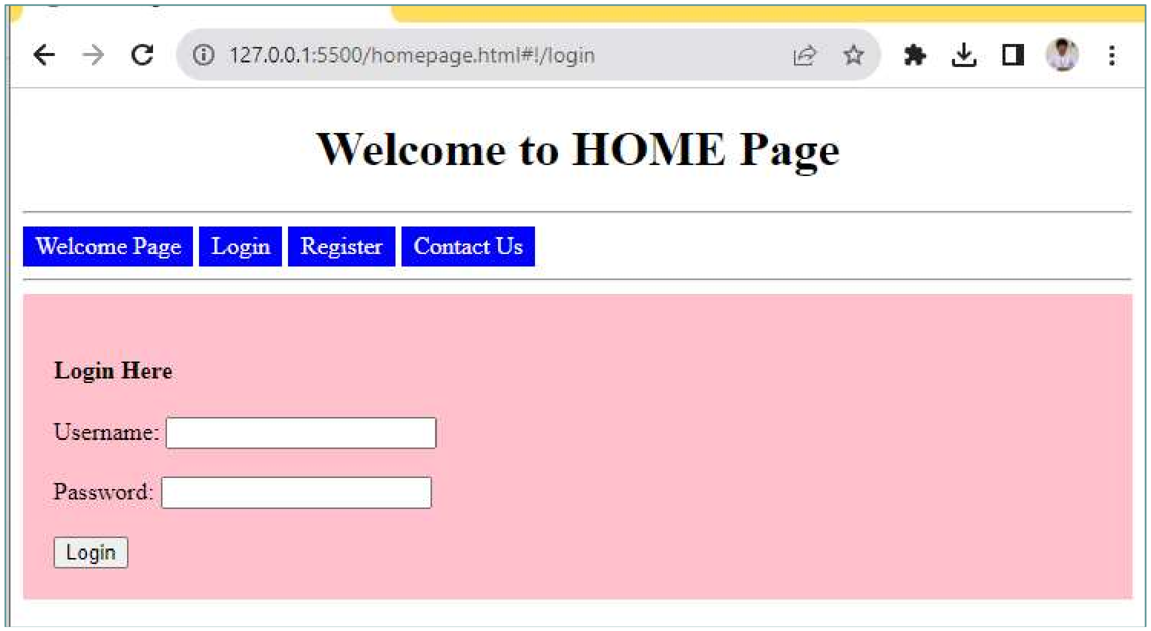
127.0.0.1:5500/homepage.html#!/

Welcome to HOME Page

Welcome Page | Login | Register | Contact Us

Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
MoE, UGC & AICTE Approved
NAAC A++ Accredited
Academics
We provide the best infrastructural facilities and a rich learning experience to educate, engage and empower our students to be engineers, technologists, scientists, artists, professionals, managers and leaders. Programmes are offered in Engineering,

Welcome Page – When the page is first loaded in browser



127.0.0.1:5500/homepage.html#!/login

Welcome to HOME Page

Welcome Page | Login | Register | Contact Us

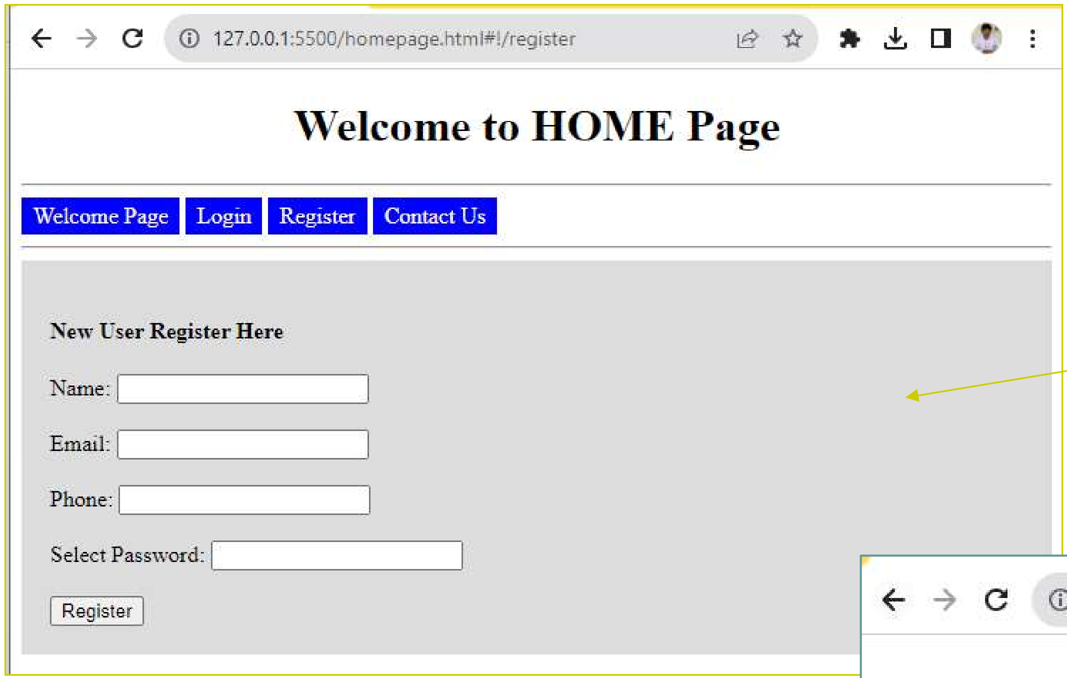
Login Here

Username:

Password:

Login

Login Page – When the hyperlink “Login” is clicked



127.0.0.1:5500/homepage.html#!/register

Welcome to HOME Page

New User Register Here

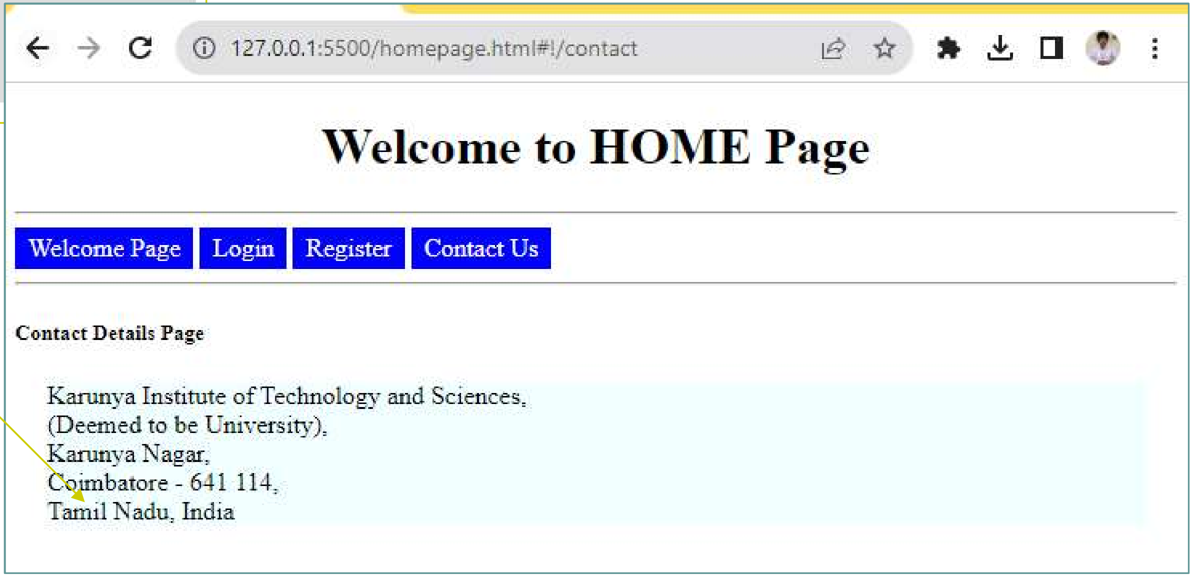
Name:

Email:

Phone:

Select Password:

Register Page – When the hyperlink "Register" is clicked



127.0.0.1:5500/homepage.html#!/contact

Welcome to HOME Page

Contact Details Page

Karunya Institute of Technology and Sciences,
(Deemed to be University),
Karunya Nagar,
Coimbatore - 641 114,
Tamil Nadu, India

Contact Page –
When the hyperlink "Contact Us" is clicked