Angular JS Introduction

- Angular JS is an open-source JavaScript framework.
- It is client-side JavaScript framework.
- It is built by Google for creating dynamic web applications.
- It can be freely used, changed and shared by anyone.
- It is a framework for building single page applications.
- Basic knowledge requirement for Angular JS is HTML + CSS + JavaScript.
- AngularJS code can be added to an HTML page with a <script> tag.
- AngularJS extends HTML attributes with Directives and binds data to HTML with Expressions.
- AngularJS is distributed as a JavaScript file and can be added to a web page with a script tag.

Characteristics/Features:

1. Data Binding

- In an angular application, we don't need to write separate code to perform the data binding functionality.
- By adding some code, we can easily bind data from HTML control to application data.
- Any extra code is not written to bind with HTML control.

2. MVC Architecture

- It is built using MVC architecture that stands for Model View and Controller.
- It separates the application into three parts model part, view part and controller part as per the components of MVC architecture.
- Using this, architecture presentation part, logic part and application data part is split into the separate section which allows managing of application in easy manner.

3. Directives

- View of AngularJS, mix data from a model into Hyper Text Markup Language templates. Angular JS directives are used for the same purpose.
- It tells how to combine data into the HTML template.
- With the use of directive, we can provide extra functionality to our angular application.
- Angular provides a way to create custom directives too.

4. Not Browser Specific

- Angular applications are not browser specific means there is no browser constraint on an angular application.
- It can run on all major browsers except internet explorer 8.0 and smartphones including Android and IOS based phones/tablets.

5. Code Less

- A programmer can write less and perform more functionalities with the same code.
- Filters in angular provide the functionality of write less do more.
- The various filters in angular are uppercase, lowercase, currency etc.
- You can use the filter and easily format the data.

6. Speed and Performance

Speed and performance of angular are faster because of three things:

- **Code Generation** – When you are writing code using angular, it converts your template into a highly optimized code that gives you an advantage of handwritten code with the productivity of framework.
- **Universal** – The first view of your application on .net, PHP, node.js and other servers that is till now dependent on HTML CSS for their front end which can serve using angular.
- **Code Splitting** – Its new component router loads angular app quickly. It provides the ability of automatic code splitting too. Therefore, only that code is loaded which is requested to render the view.

7. Dependency Injection

- Dependency Injection specifies a design pattern in which components are given their dependencies instead of hard coding them within the component.
- Whenever, angular JS detect that you need a service then it immediately provides an instance for that.
- It allows you to ask for your dependencies rather than having to go look for them or making it by yourself.

8. Deep Linking

- It allows to bookmarks the web page.
- The page gets saved by its URL without getting its state changed.
- Whenever the request is made by a user for that page it will get displayed in the same state as before.

9. Routing

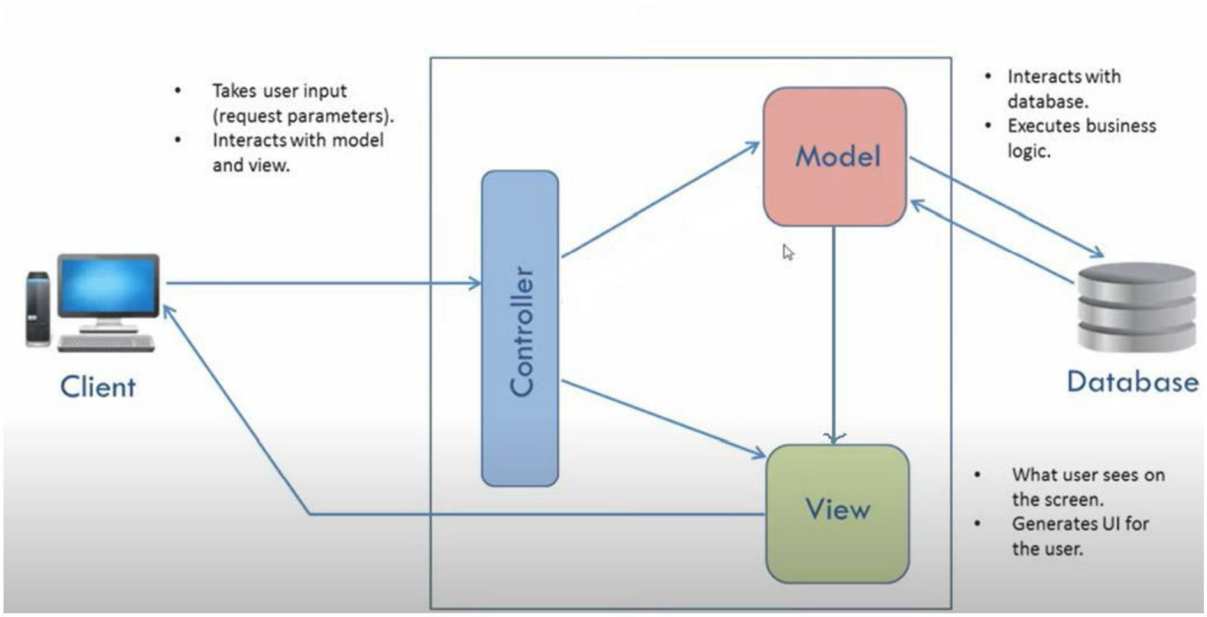
- Routing allows the switching between views.
- Being a single page application ngRoute directive provided by angular, helps a user to navigate from one view to another, but the application will remain single page.
- It means without reloading the angular application you can switch to different pages in your application.

10. Productivity

- **Template –** Template in the angular application allows a developer to create user interface quickly as it provides simple and powerful template syntax.
- **Angular CLI –** It is a command line tool. It starts building an application very fast. It adds components, tests it and then deploys it instantly.
- **IDE** – Intelligent code completion is possible through IDE. It will find instant errors and provides other feedbacks too.

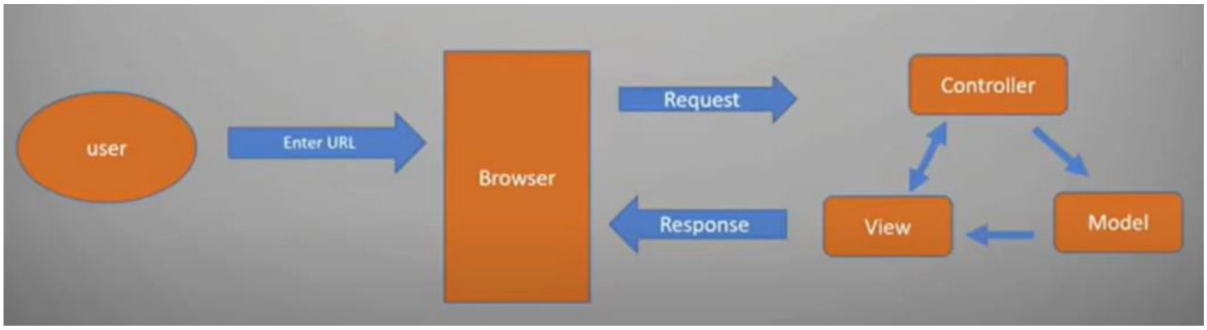
MVC architecture

- MVC stands for Model View Controller.
- It is a software design pattern for developing web applications.
- It handles internal activities separately using Model, View and Controller.



1. Client sends request to the Controller for getting all students' detail which is available in database.
2. Controller passes the request to the Model and model communicate with database and get back response.

3. Model passes the row data to the View.
4. View does proper formatting to the data as per the instruction provided by Controller so it becomes visible.
5. Client can see the formatted data provided by the View.



The MVC pattern is made up of the following three parts:

1. **Model:** It is responsible for managing application data. It responds to the requests from view and to the instructions from controller to update itself. (Using JSON Data object which is loaded inside view.). It can be represented as body of code which is used to represent the data.
2. **View:** It is responsible for displaying all data or only a portion of data to the users. It also specifies the data in a particular format triggered by the controller's decision to present the data. They are script-based template systems such as JSP, ASP, PHP and very easy to integrate with AJAX technology. (Using HTML Tags.). It is a body of code which is used to represent UI. Application can have multiple views. Each view represents some portion of your model.
3. **Controller:** It is responsible to control the relation between models and views. It responds to user input and performs interactions on the data model objects. The controller receives input, validates it, and then performs business operations that modify the state of the data model. (Using JS functions (logic / Data loading).)

Setting up Environment

Follow following steps to create AngularJS web application:

1. Create one folder (Ex: AngularJS-App).

2. Create one .html file in same folder (Ex: index.html). Write basic HTML tags in the file.
3. Open “angularjs.org”.
4. Click on “Download angularjs”. You will get follow:

5. There is a need to include AngularJS library using <script> element. If it is not specified then the AngularJS related code cannot be processed. For doing it:

   Copy CDN (Content Delivery Network) path and set this path with <script> tag in <head> tag. (With internet connectivity for accessing file)

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

   OR

   Download angular.min.js file from this location. And save this file in created folder (i.e. in AngularJS-App). Then this file can be locally accessed without internet connection.

```
<script src='angular.min.js'></script>
```

6. Write angularJS according code and run the index.html file in any web browser.

Angular JS Directives

- AngularJS uses directives to augment HTML with extra functionality.
- Directives are a convenient way to declaratively call JavaScript functions.
- Directives are markers on a DOM element that tell AngularJS to attach a specified behaviour to that DOM element or even transform the DOM element and its children. In short, it extends the HTML.
-

- Most of the directives in AngularJS are starting with ng- where ng stands for Angular. AngularJS includes various built-in directives.

➢ ng-app

- It is used to define AngularJS application.
- It is used to inform our page that it is under the control of AngularJS.
- It also specifies the Parent element where the AngularJS code is written.
- If ng-app is specified with <body> tag then it is interpreted that under the boundary of BODY tag, code is under the control of AngularJS.
- If ng-app is specified within another tag (i.e.) then it is interpreted that under the boundary of this tag, the code is under the control of AngularJS. But outside its boundary, the code is not under the control of AngularJS. (Ex: p1.html file)
- If ng-app is specified with <html> tag then it is interpreted that the AngularJS code needs to execute for this html file. (Ex: p2.html file)
- Example:

p1.html file:----------

```
<!DOCTYPE html>
<html>
<head>
    <title>Angular tutorial</title>
    <!-- <script
src='https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js'></s
cript> -->
    <script src='angular.min.js'></script>
</head>
<body>
    {{"Hello"}}
    <div ng-app>
    {{3+5}}
    </div>
</body>
</html>
```

Output:

- Example:

p2.html file:----------

```
<!DOCTYPE html>
<html ng-app>
<head>
    <title>Use of ng-app directive</title>
    <script src='angular.min.js'></script>
</head>
<body>
    {{"Hello"}}
    <div>
    {{3+5}}
    </div>
</body>
</html>
```

Output:

Hello
8

Data Binding:

- Data binding binds the expression with data.
- The data which is written within {{…}} is known as AngularJS expression.
- When we use ng-model/bind=”variable_name” then the respective data is bind with mentioned variable_name.

➢ ng-model

- The ng-model directive binds the value of HTML controls (input, select, textarea) to application variable name.
- The value which is inserted using above mentioned controls are bind in specific variable using ng-model directive.
- The ng-model directive is used for two-way data binding. It transfers the data from controller to view and vice-versa. (Two way data binding)
- It can also
    - Provide type validation for application data (no, email, required)
    - Provide status for application data (invalid, error)
    - Provide CSS classes for HTML elements.
    - Bind HTML elements to HTML forms.
- Ex:

p3.html file:----------

```
<!DOCTYPE html>
<html ng-app>
<head>
    <title>Use of ng-model directive</title>
    <script src='angular.min.js'></script>
</head>
<body>
    <label>Enter Student's surname:
    <input type="text" ng-model="stsname"/><br/>
    <label>Enter student's name:
    <input type="text" ng-model="stname"/>
    <h2>Student's name is {{stsname+" "+stname}}</h2>
</body>
</html>
```

p4.html file:----------

```
<!DOCTYPE html>
<html ng-app>
<head>
    <title>Use of ng-model directive</title>
    <script src='angular.min.js'></script>

</head>
<body>
    <form className='formmargin'>
        <fieldset>
        <legend >Student's personal information</legend>  <br/>
       <label>Name: (Surname first)</label><br/>
       <input type="text" ng-model="stname"><br/>
        <label>Enter your password</label><br/>
        <input type="password" ng-model="stpass"><br/> <br/>
        <label>Enter your email</label><br/>
        <input type="email" ng-model="stmail"><br/>
        Enter your Address:<br/>
        <textarea ng-model="stadd"></textarea><br/> <br/>
        <label>Enter your gender</label><br/>
        <input type="radio" id="gender" value="male" ng-model="stgender">Male

        <input type="radio" id="gender" value="female" ng-
model="stgender">Female
        <input type="radio" id="gender" value="others" ng-
model="stgender">others <br/>

        <label>Select couse: </label>
        <select ng-model="stcourse">
```

```
            <option value="BCA">BCA</option>
            <option value="BBA">BBA</option>
            <option value="B.Com">B.Com</option>
        </select><br/><br/>

        <label>Select Hobbies:</label><br/>
        <input type="checkbox" value="Drawing" ng-model="stdraw">Drawing<br/>
        <input type="checkbox" value="Singing" ng-model="stsing">Singing<br/>
        <input type="checkbox" value="Dancing" ng-model="stdance">Dancing<br/>

      </fieldset>
    </form>
    <h1>Student's detail</h1>
    <h2>Name: {{stname}}</h2>
    <h2>Password: {{stpass}}</h2>
    <h2>Email: {{stmail}}</h2>
    <h2>Address: {{stadd}}</h2>
    <h2>Gender: {{stgender}}</h2>
    <h2>Course: {{stcourse}}</h2>
    <h2>Hobbies: </h2>
    <h2>Drawing: {{stdraw}}</h2>
    <h2>Singing: {{stsing}}</h2>
    <h2>Dancing: {{stdance}}</h2>
</body>
</html>
```

➢ Ng-bind

- The ng-bind directive binds the data to the HTML elements.
- This directive binds the content of specific element (i.e. <p>, , etc.) to the application variable name.
- It transfers data from Controller to view. View to controller is not possible. (one way data binding)

P5.html file:----------

```
<!DOCTYPE html>
<html ng-app>
<head>
    <title>Use of ng-bind directive</title>
    <script src='angular.min.js'></script>
</head>
<body>
    <label>Enter Student's surname:
    <input type="text" ng-model="stsname"/><br/>
    <label>Enter student's name:
    <input type="text" ng-model="stname"/>
```

```
    <h2 ng-bind="stsname"></h2>
    <h2 ng-bind="stname"></h2>
</body>
</html>
```

➢ ng-init

- It is used to initialize the variable.
- Syntax: ng-init="var_name1=value;var_name2=value;..."

P6.html file: ----------

```
<!DOCTYPE html>
<html ng-app>
<head>
    <title>Use of ng-init directive</title>
    <script src='angular.min.js'></script>
</head>
<body ng-init="stname='Het';stsname='Patel'">
    <label>Enter Student's surname:
    <input type="text" ng-model="stsname"/><br/>
    <label>Enter student's name:
    <input type="text" ng-model="stname"/>
    <h2>Student's name is {{stsname+" "+stname}}</h2>
</body>
</html>
```

p7.html file:----------

```
<!DOCTYPE html>
<html ng-app>
<head>
    <title>Use of ng-model and ng-init directive</title>
    <script src='angular.min.js'></script>

</head>
<body style="background-color:{{effect}}" ng-init="textclr='yellow'">
    <input type="text" style="background-color: {{textclr}};" ng-
model="effect"/>
    <input type="text" ng-model="clr" style="background-color: {{textclr}};"/>
    <h2 style="color:{{clr}}">Use of Expression</h2>
</body>
</html>
```

Expressions (Numbers, String, Objects, Array)

- AngularJS expression is like JavaScript expression which is written within {{...}}.
- Syntax: {{expression}}
- AngularJS evaluates the specified expression and binds the result data to HTML.
- It includes literals, operators and variables.
- AngularJS expression cannot contain conditions, loops, exceptions or regular expressions
- AngularJS expression cannot declare functions.
- AngularJS expression cannot contain comma or void.
- AngularJS expression cannot contain return keyword.
- It transfers data from Controller to View. View to controller is not possible using it. (One-way data binding)

Numbers:

- Ex: {{5+7}} Output:12
- Ex: {{no*5}} Output: (if no=5 then 25)

String:

- Strings are written with double quotes.
- To combine string + operator is used.
- Ex: {{"Web"+" " + "Designing"}} Output: Web Designing

Object:

- The object can also be accessed in expression.
- Ex: {{stud.m1 +stud.m2}}

Array:

- Array can also be initialized and accessed using expression.
- Array index starts with 0.
- Ex: {{marks[4]}} It returns the 5th subject's marks.

p8-expression.html file:----------

```
<!DOCTYPE html>
<html ng-app>
<head>
    <title>Expression</title>
    <script src='angular.min.js'></script>

</head>
<body>
    <h1>Expression</h1>
    <h2><u>Number</u></h2>
    <h3>{{3+5}}</h3>
    <h3 ng-init="n1=3;n2=50">{{n1*n2}}</h3>

    <h2><u>String</u></h2>
    <h3>{{"Web"+"Designing"}}</h3>
    <h3 ng-init="s1='Patel';s2='Vraj'">{{s1+" "+s2}}</h3>

    <h2><u>Object</u></h2>
    <div ng-init="stud={rno:101,name:'Raju',m1:34,m2:45}">  </div>
    <h3>Rollno: {{stud.rno}}</h3>
    <h3>Name: <span ng-bind="stud.name"></span></h3>
    <h3>Marks: {{stud.m1+stud.m2}}</h3>

    <h2><u>Array</u></h2>
    <div ng-init="marks=[23,34,45,34,23,100]">  </div>
    <h3>1st subject's Marks: {{marks[0]}}</h3>
    <h3>2nd subject's Marks: {{marks[1]}}</h3>
    <h3>3rd subject's Marks: {{marks[2]}}</h3>
    <h3>4th subject's Marks: {{marks[3]}}</h3>
    <h3>5th subject's Marks: {{marks[4]}}</h3>
    <h3>Practical exam's Marks: <span ng-bind="marks[5]"> </span></h3>


</body>
</html>
```

Angular JS Directives

➢ ng-show

- ng-show is used to display specific data as per mentioned condition.
- Ng-show evaluates Boolean values only. The expression which returns true or false can be evaluated by this directive.
- This directive is used when we want to show any specific UI region as per user's input or condition.

p9.html file:----------

```
<!DOCTYPE html>
<html ng-app>
<head>
    <title>Use of ng-model directive</title>
    <script src='angular.min.js'></script>
</head>
<body>
    <p ng-show="true">Display</p>
    <p ng-show="false">No Display</p>

    <label>Enter Student's age:
    <input type="text" ng-model="age"/><br/>

    <p ng-show="age>=18">
        <label>Enter Student's weight:
        <input type="text" ng-model="weight"/><br/>
        <p ng-show="weight>=50">Eligible to donate blood</p>
    </p>
</body>
</html>
```

➢ ng-hide

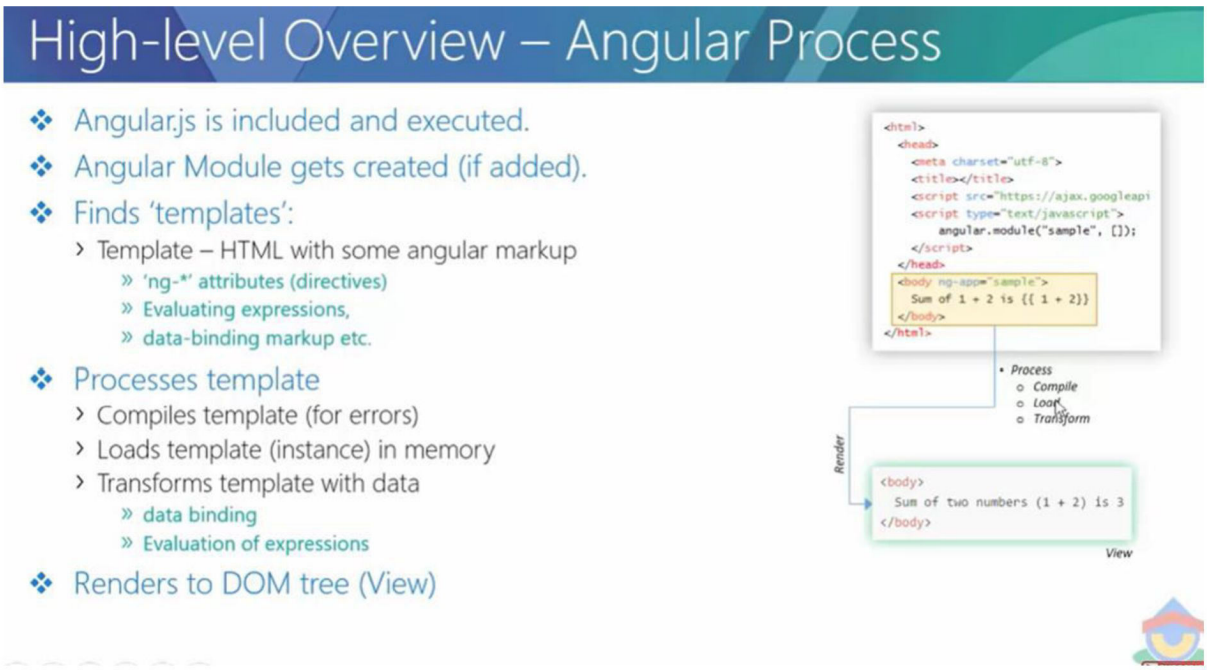
- ng-hide is used to hide specific data as per mentioned condition.
- Ng-hide evaluates Boolean values only. The expression which returns true or false is evaluated by this directive.
- This directive is used when we want to hide any specific UI region as per user's input or condition.

p10.html file:----------

```
<html lang="en" ng-app>
<head>
    <title>Use of ng-hide directive</title>
    <script src='angular.min.js'></script>
</head>
<body ng-init="gender='male'">
    <p ng-hide="true">Display</p>
    <p ng-hide="false">No Display</p>
    <input type="radio" value="male" ng-model="gender" name="gender"/>Male
    <input type="radio" value="female" ng-model="gender" name="gender"/>Female
    <div  ng-hide="gender=='male'">
        select your choice:
        <input type="checkbox" value="NSS" ng-model="nss">NSS
        <input type="checkbox" value="NCC" ng-model="ncc">NCC
```

```
        <p> NSS: {{nss}}<br/>
            NCC: {{ncc}}</p>

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



Module:

- Module is a container for the different part of the application. It includes controllers, filters, directives etc.
- One separate JavaScript file needs to create for declaration of Module and for configuration module.
- This JS file needs to load with <script src="file_name"> tag in <head> part.
- Controller JS function is written within this module.
- Create module:

  var mod_instance= angular.module("mod_name",[dependency]);

  Ex: var myApp=angular.module("myModule",[]);
- Loading module in HTML file:

  ng-app="mod_name"

  Ex: <html ng-app="myModule">

Controller:

- Controller is the JavaScript object (function) which contains application logic.
- It allows us to send or receive data between views and application logic.

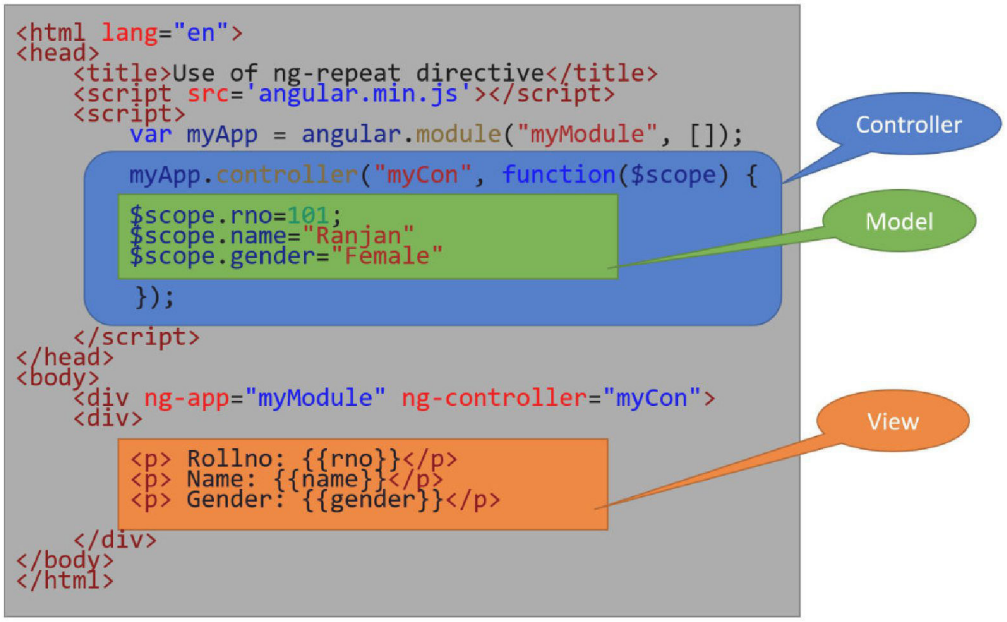
- The created controller is attached to DOM elements using ng-controller directive.
- It is used to build a model for the view to display.
- It is defined within Module.
- It is attached with module using created module instance.
- Creating Controller:

  mod_instance.controller("con_name",function(){...block_code...});

  Ex: myApp.controller("myController",function($scope){
  $scope.sub="Advanced Web Desigining";
  });
- Accessing Controller in HTML file:

  <... ng-controller="con_name">

  Ex: <h2 ng-controller="myController"> {{sub}}




```
<html lang="en">
<head>
    <title>Use of ng-repeat directive</title>
    <script src='angular.min.js'></script>
    <script>
        var myApp = angular.module("myModule", []);
        myApp.controller("myCon", function($scope) {
        $scope.rno=101;
        $scope.name="Ranjan"
        $scope.gender="Female"
        });
    </script>
</head>
<body>
    <div ng-app="myModule" ng-controller="myCon">
    <div>
        <p> Rollno: {{rno}}</p>
        <p> Name: {{name}}</p>
        <p> Gender: {{gender}}</p>
    </div>
</body>
</html>
```

p11-module-controller.html file:----------

```
<html lang="en">
<head>
    <title>Use of ng-repeat directive</title>
    <script src='angular.min.js'></script>
    <script>
        var myApp = angular.module("myModule", []);
        myApp.controller("myCon", function($scope) {
        $scope.rno=101;
        $scope.name="Ranjan"
        $scope.gender="Female"
```

```
  });

    </script>
</head>
<body>
    <div ng-app="myModule" ng-controller="myCon">
    <div>
        <p> Rollno: {{rno}}</p>
        <p> Name: {{name}}</p>
        <p> Gender: {{gender}}</p>
    </div>
</body>
</html>
```

- Controller JavaScript function maintains the application data and behaviour using $scope object.
- Separate script file for controller and module can also be created and the same file can be used in HTML file as reference using “src” attribute in <script> tag.

p12.js file:----------

```
var myApp=angular.module("myModule",[]);
myApp.controller("myCon",function($scope){
    $scope.sub="Advanced Web Designing";
        });
```

p12-module-controller.html file: ----------

```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<script src="p12.js"></script>
<body>

<div ng-app="myModule" ng-controller="myCon">
<b><h1>{{ sub }}</h1></b>
</div>
</body>
</html>
```

$scope:

- $scope in an AngularJS is a built-in object, which contains application data and methods.
- We can create properties to a $scope object inside a controller function and assign a value or function to it.
- The $scope is glue between a controller and view (HTML).
- $scope is used to represent MODEL part of MVC.
- It transfers data from the controller to view and vice-versa.
- we can attach properties and methods to the $scope object inside controller function.
- AngularJS creates and injects a different $scope object to each controller in an application. So, the scope of $scope is local for specific Controller. The data and methods attached to $scope inside one controller cannot be accessed in another controller.
- With the nested controller, child controller will inherit the parent controller's scope object. Therefore, child controller can access properties added in parent controller but parent controller cannot access properties added in child controller.
- The view can display $scope data using an expression, ng-model, or ng-bind directive.

p13-scope.js file:----------

```
var myApp=angular.module("myModule",[]);
myApp.controller("myCon",function($scope){
    var student=["Krishna","Madhav", "Murlidhar", "Kishan","Girdhar","Gopal"];
    var course=    [{cno:"c1", cname:"BCA",duration:"3 years", sem:6},
                    {cno:"c2", cname:"BBA",duration:"3 years", sem:6},
                    {cno:"c3", cname:"BCom",duration:"3 years", sem:6},
                    {cno:"c4", cname:"HonorsBCA",duration:"1 year", sem:2}];
    $scope.stud=student;
    $scope.course_detail=course;
        });
```

p13-scope.html file:----------

```
<!DOCTYPE html>
<html>
<head>
<script src="angular.min.js"></script>
<script src="p13-scope.js"></script>
</head>
<body>
    <div ng-app="myModule" ng-controller="myCon">
        <h1> <u>View of $scope:</u></h1>
```

```
        <h2>Using Expression:</h2><h3> {{ stud[0]}}</h3>
        <h2>Using ng-bind:</h2>
        <h3 ng-bind="stud[1]"></h3>
        <h2>Using ng-model:</h2>
        <input type="text" ng-model="stud[2]"/>
    </div>
</body>
</html>
```

➢ ng-repeat

- ng-repeat directive is used to repeat the task n numbers of time.
- The ngRepeat directive will repeat the instance of the element upon which it is declared for each item in a collection. (In below example, <li> tag repeats based on the values available in list.
- Ex:

p14-repeat.html file:----------

```
<html lang="en" ng-app>
<head>
    <title>Use of ng-repeat directive</title>
    <script src='angular.min.js'></script>
</head>
<body ng-app="myApp" ng-init="marks=[12,34,45,56]">

    {{"Hello"}}
    <div>
        <ol>
            <li ng-repeat="m in marks">{{m}}</li>
        </ol>
    </div>
</body>
</html>
```

- Display array using ng-repeat

p15-repeat.html file:----------

```
<html>
<head>
    <title>Use of ng-repeat directive</title>
    <script src='angular.min.js'></script>
    <script src="p13-scope.js"></script>
</head>
<body>
```

```
    <div ng-app="myModule" ng-controller="myCon">
        <h1> <u>Total no. of students: </u>{{stud.length}}</h1>
        <h1> <u>List of students: </u></h1>
        <div>
            <ol>
            <li ng-repeat="m in stud">{{m}}</li>
        </ol>
    </div>
</body>
</html>
```

- Display documents using ng-repeat

p16-repeat.html:----------

```
<html>
<head>
    <title>Use of ng-repeat directive</title>
    <script src='angular.min.js'></script>
    <script src="p13-scope.js"></script>
</head>
<body>
    <div ng-app="myModule" ng-controller="myCon">
        <h1> <u>Total no. of courses: </u>{{course_detail.length}}</h1>
        <h1> <u>List of Courses: </u></h1>
        <div>
            <ol>
            <li ng-repeat="m in course_detail">Cno={{m.cno}},Course
name={{m.cname}}, Duration:{{m.duration}}, Total Sem: {{m.sem}}</li>
        </ol>
    </div>
</body>
</html>
```

➢ ng-class

- The ng-class Directive is used to specify the CSS classes on HTML elements.
- It is used to dynamically bind classes on an HTML element.
- The value of the ng-class directive can be a string, an object, or an array.
    - If it is a string, it should contain one or more, space-separated class names.
    - As an object, it should contain key-value pairs, where the key is the class name of the class you want to add, and the value is a boolean value.
    - The class will only be added if the value is set to true.

- As an array, it can be a combination of both. Each array element can be either a string, or an object, described as above.

p17-class.html file: ----------
Change background color and display shape as per user's choice:

```
<html ng-app>
<head>
    <title>Use of ng-repeat directive</title>
    <script src='angular.min.js'></script>
    <style>
        .red{background-color: red;}
        .green{background-color: green;}
        .blue{background-color: blue;}
        .gray{background-color: gray;}
        .square{
            width: 80px;
            height: 80px;
            background-color: azure;

        }
        .rectangle{
            width: 160px;
            height: 80px;
            background-color: azure;
        }
        .oval{
            width: 160px;
            height: 80px;
            background-color: azure;
            border-radius: 100px / 50px;
        }
    </style>
</head>
<body ng-class="clr">
    <div>
        <b>Select the color:</b>
        <input type="radio" id="clr" ng-model="clr" value="red">Red
        <input type="radio" id="clr" ng-model="clr" value="green">Green
        <input type="radio" id="clr" ng-model="clr" value="blue">Blue
        <input type="radio" id="clr" ng-model="clr" value="gray">Gray <br/>
        <b>Select Shape:</b>
        <input type="radio" ng-model="shape" value="square"> Square
        <input type="radio" ng-model="shape" value="rectangle"> Rectangle
        <input type="radio" ng-model="shape" value="oval"> Oval

        <div ng-class="shape"></div>
    </div>
```

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

- Ng-class can be used to apply css as per mentioned condition.

p18-class.html file:----------

```
<html>
<head>
    <title>Use of ng-repeat directive</title>
    <script src='angular.min.js'></script>
    <script src="p13-scope.js"></script>
    <style>
        .red{color: red;}
        .green{color: green;}
        .blue{color: blue;}

    </style>
</head>
<body>
    <div ng-app="myModule" ng-controller="myCon">
        <h1> <u>List of Courses: </u></h1>
        <div>
            <ol>
            <li ng-repeat="m in course_detail" ng-
class={"red":m.sem===6,"green":m.sem===2}><b>{{m.cname}}</b></li>
        </ol>
    </div>
</body>
</html>
```

➢ ng-animate

- To give animation through AngularJS, there is a need to use angular-animate.js file.
- ngAnimate module:
  It allows to apply animation so this module can be used as dependency if we are using another module otherwise it can be written with ng-app directive.
  - Directly: ng-app=”ngAnimate”
  - Through module:
    ng-app=”myModule”……..
    Module:----------
    var myApp = angular.module('myModule', ['ngAnimate']);

  It adds and removes classes.

It does not animate your HTML elements, but when ngAnimate notice certain events, like hide or show of an HTML element, the element gets some pre-defined classes which can be used to make animations.

The directives in AngularJS who add/remove classes are:

- ng-show
- ng-hide
- ng-class
- ng-view
- ng-include
- ng-repeat
- ng-if
- ng-switch

- Animated effect can be given using some directives like ng-hide, ng-show.
- The required effect can be set using CSS and which can be implemented using mentioned directives.

p19-animate.html

```
<!DOCTYPE html>
<html ng-app="ngAnimate">
<head>
    <title>Animation</title>
    <script src='angular.min.js'></script>
    <script src='angular-animate.js'></script>
    <style>
        div{
            transition:linear 0.5s;
            background-color: red;
            color:aliceblue;
            height: 100px;
            width:100%;
            top:0;
            left:0;
        }

        .ng-hide{
            background-color:transparent;
            height: 0;
            width:0;
            top: -200px;;
            left: 200px;;
        }
        .ng-show{
            background-color:blue;
            color:antiquewhite;
            height: 10px;
```

```
            width:10px;
            top: 0;
            left: 0;
        }
    </style>
</head>
<body>
    <label>Hide the block:
        <input type="checkbox" ng-model="myCheck1"/><br/>
        <div ng-hide="myCheck1"><h1>Animation using ng-hide</h1></div>
    <br>
    <label>Show the block:
        <input type="checkbox" ng-model="myCheck2"/><br/>
        <div ng-show="myCheck2"><h2>Animation using ng-show</h2></div>
</body>
</html>
```

Angular JS Filters

- Filters are used to format the data.
- Filters are generally added to the expressions by using "|" symbol.
- Syntax: {{variable | filter}}
- Some of the filters are discussed below:
    - uppercase: It is used to convert the given text into uppercase.
    - lowercase: It is used to covert the given text into lowercase.
    - currency: It is used to convert the given numbers into currency format.
    - order by: The list is ordered by expression.

p20-filters.html file:----------

```
<html>
<head>
    <title>Use of ng-repeat directive</title>
    <script src='angular.min.js'></script>
    <script src="p13-scope.js"></script>

</head>
<body>
    <div ng-app="myModule" ng-controller="myCon" ng-init="purchase_amt=23000">
        <h1> <u>List of Courses in order: </u></h1>

        <div>
        <ol>
            <li ng-repeat="m in course_detail |orderBy:'cname'"><b>Course
Name:{{m.cname| lowercase}}   duration:{{m.duration|uppercase}}</b></li>
        </ol>
```

```
        <h2>Purchase Amount: {{purchase_amt|currency}}</h2>
    </div>
</body>
</html>
```

Angular JS Routing:

- The ngRoute module helps in developing Single Page Application.
- To access different pages within webpage with Single Page Application feature, without reloading each page then Routing is used.
- It is possible using ngRoute module.
- The ngRoute module routes your application to different pages without reloading the entire application.
- "angular-route.js" file is required for implementing Routing concept.
- ngRoute can be added as dependency in module.
  Ex: var myApp = angular.module("myModule", ["ngRoute"]);
- ngRoute module provides "$routeProvider":
  - It helps in configuring different files (routes) within application. It is used to define which page you want to display when user click on any link.
  - It has config method. Work registered in the config method will be performed when the application is loading.
  - It accepts either when() or otherwise() method.

- To access these files, application requires container named "ng-view directive" which displays the content of mentioned files.
- ng-view can be declared as:
  - Using element: <div ng-view>
  - Using ng-view tag: <ng-view></ng-view>
  - Using class:

- Two properties can be used to link with specific file or code:
  - templateUrl property: Using this property directly the URL of specific file is mentioned within application to access the code of the file.
  - template property: Using this property, directly the code block is executed at specified location.

```
<!DOCTYPE html>
<html>
<head>
        <script src=" angular.min.js">
        </script>
        <script src="angular-route.js">
        </script>
</head>

<body ng-app="myApp">
        <p>
                <a href="#/!">
                        <img src=" geeksforgeeks-logo1.png"
                                alt="GeeksforGeeks"
                                style="width: 90vw;">
                </a>
        </p>
        <a href="#!courses">Courses@geeksforgeeks</a>
        <br>
        <a href="#!internships">Internships@geeksforgeeks</a>
        <div ng-view></div>

        <script>
                const app = angular.module("myApp", ["ngRoute"]);
                app.config(function ($routeProvider) {
                        $routeProvider
                                .when("/", {
                                        template: `<h1>Welcome to
GeeksForGeeks</h1>
                                        <p>
                                                Click on the links to change this content
                                        </p>`
                                })
                                .when("/courses", {
                                        template: `<h1>Courses Offered</h1>
                                        <p>
                                                <ul>
                                                <li>Machine Learning Foundation</li>
                                                <li>Geeks Classes</li>
                                                <li>System Design</li>
                                                </ul>
                                        </p>`
```

```
                        })
                        .when("/internships", {
                                template: `<h1>Hire With Us</h1>
                                <p>
                                        <ul>
                                        <li>Software Developer</li>
                                        <li>Technical Content Writer</li>
                                        <li>Technical Content Engineer</li>
                                        </ul>
                                </p>`
                        });
                });
        </script>
</body>
</html>
```

Navigating another file using SPA feature:

```
<body ng-app="myApp">
```

```
<p><a href="#/!">Main</a></p>

<a href="#!red">Red</a>
<a href="#!green">Green</a>
<a href="#!blue">Blue</a>

<div ng-view></div>

<script>
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
  $routeProvider
  .when("/", {
    templateUrl : "main.htm"
  })
  .when("/red", {
    templateUrl : "red.htm"
  })
  .when("/green", {
    templateUrl : "green.htm"
  })
  .when("/blue", {
    templateUrl : "blue.htm"
  });
});
</script>
</body>
```

Forms (Events, Data validation, ng-click):

- Forms in AngularJS provides data-binding and validation of input controls.
- Input Controls are the HTML input elements:
  - input elements
  - select elements
  - button elements
  - textarea elements
- Input controls provides data-binding by using the ng-model directive. This directive binds the input controller to the rest of your application.

Create form and display students' detail:

```
<html lang="en">
<head>
    <title>Use of form</title>
    <script src='angular.min.js'></script>
</head>
```

```
<body>
    <div ng-app="myApp" ng-controller="formCtrl">
        <form novalidate>
                 <fieldset>
                 <legend >Student's personal information</legend>  <br/>
                <label>Name: (Surname first)</label><br/>
                <input type="text" ng-model="user.fname"><br/>
                 <label>Enter your password</label><br/>
                 <input type="password" ng-model="user.fpw"><br/> <br/>
                 <label>Enter your email</label><br/>
                 <input type="email" ng-model="user.femail"><br/>
                 Enter your Address:<br/>
                 <textarea ng-model="user.fadd"></textarea><br/> <br/>
                 <label>Enter your gender</label><br/>
                 <input type="radio" ng-model="fgender" value="male"/>Male
                 <input type="radio" ng-model="fgender" value="female"/>Female
                 <input type="radio" ng-model="fgender" value="others"/>others
                 <br/>
                 <label>Select couse: </label>
                 <select ng-model="user.fcourse">
                    <option value="BCA">BCA</option>
                    <option value="BBA">BBA</option>
                    <option value="B.Com">B.Com</option>
                 </select><br/><br/>
                 <button ng-click="reset()">Reset</button>
                </fieldset>
              </form>
              <div>
                user: {{user}}<br/>
                master: {{master}}<br/>
                Name: {{user.fname}}<br/>
                Password: {{ user.fpw}}<br/>
                email id: {{ user.femail}}<br/>
                Address: {{ user.fadd}}<br/>
                Gender: {{ user.fgender}}<br/>
                Course: {{ user.fcourse}}<br/>


              </div>
        </div>


        <script>
        var app = angular.module('myApp', []);
        app.controller('formCtrl', function($scope) {
          $scope.master = {fname: "", fpw: "",femail: "", fadd: "", fgender:
"male",fcourse: "BCA"};
          $scope.reset = function() {
            $scope.user = angular.copy($scope.master);
```

```
        };
        $scope.reset();
    });
    </script>

</body>
</html>
```

Form Validation:

- 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.
- Client-side validation cannot alone secure user input. Server-side validation is also necessary.
- You can use standard HTML5 attributes to validate input, or you can make your own validation functions.
- HTML5 attributes:
    - required: to specify that the input field must be filled out.
    - email: to specify that the value must be an e-mail. etc.
    - maxlength: to specify the maximum length.
    - min: to specify minimum number.
    - max: to specify maximum number.

- "novalidate" keyword is used when you don't want to validate form.

- Form input fields have the following states: (returns true or false)
    - $untouched: It shows that field has not been touched yet.
    - $touched: It shows that field has been touched.
    - $pristine: It represents that the field has not been modified yet.
    - $dirty: It illustrates that the field has been modified.
    - $invalid: It specifies that the field content is not valid.
    - $valid: It specifies that the field content is valid.

- Forms have the following states: (returns true or false)
    - $pristine: It represents that the fields have not been modified yet.
    - $dirty: It illustrates that one or more fields have been modified.
    - $invalid: It specifies that the form content is not valid.
    - $valid: It specifies that the form content is valid.
    - $submitted: It specifies that the form is submitted.

- These states returns true or false which can help for showing appropriate message to user.

Checking of states:

```
<html lang="en">
<head>
    <title>Use of form</title>
    <script src='angular.min.js'></script>
</head>
<body>
    <div ng-app>
        <form name="frm">
                <label>Name:</label>
                <input type="text" ng-model="fname" name="fname"required>
                <span ng-show="frm.fname.$invalid">Wrong input</span>
<br/><br/>
                <label>Surname:</label>
                <input type="text" ng-model="sname" name="sname"
required><br/> <br/>
                <table border="1">
                  <tr>
                    <td>States</td>
                    <td>Form</td>
                    <td>Name field</td>
                  </tr>
                  <tr>
                    <td>$touched</td>
                    <td>{{frm.$touched}}</td>
                    <td>{{frm.fname.$touched}}</td>
                  </tr>
                  <tr>
                    <td>$untouched</td>
                    <td>{{frm.$untouched}}</td>
                    <td>{{frm.fname.$untouched}}</td>
                  </tr>
                  <tr>
                    <td>$pristine</td>
                    <td>{{frm.$pristine}}</td>
                    <td>{{frm.fname.$pristine}}</td>
                  </tr>
                  <tr>
                    <td>$dirty</td>
                    <td>{{frm.$dirty}}</td>
                    <td>{{frm.fname.$dirty}}</td>
                  </tr>
                  <tr>
```

```
                    <td>$valid</td>
                    <td>{{frm.$valid}}</td>
                    <td>{{frm.fname.$valid}}</td>
                  </tr>
                  <tr>
                    <td>$invalid</td>
                    <td>{{frm.$invalid}}</td>
                    <td>{{frm.fname.$invalid}}</td>
                  </tr>
                </table>
              </form>
        </div>
</body>
</html>
```

Custom validation:

- To create your own validation function is a bit trickier; You have to add a new directive to your application, and deal with the validation inside a function with certain specified arguments.

Apply custom validation to check age>=18:

```
<!DOCTYPE html>
<html>
<head>
  <title>AngularJS Form Validation</title>
  <script src="angular.min.js"></script>
</head>
<body ng-app="developapp">
  <h3>AngularJS Custom Form Validation</h3>
  <form name="form1">
    Username: <input name="username" required><br><br>
    Age:<input name="userage" ng-model="userage" required app-directive>
      <span ng-hide="form1.userage.$valid">Age >= 18.</span>
  </form>
  <script>
    var app = angular.module('developapp', []);
    app.directive('appDirective', function () {
      return {
        require: 'ngModel',
        link: function (scope, element, attr, mCtrl) {
          function myValidation(value) {
            if (value >= 18) {
              mCtrl.$setValidity('charE', true);
            } else {
              mCtrl.$setValidity('charE', false);
            }
```

```
            return value;
          }
          mCtrl.$parsers.push(myValidation);
        }
      };
    });
  </script>
</body>
</html>
```

ng-disabled:

- The ng-disabled directive sets the disabled attribute of a form field (input, select, or textarea).
- The form field will be disabled if the expression inside the ng-disabled attribute returns true.
- The ng-disabled directive is necessary to be able to shift the value between true and false.

Apply disable effect on text1 and text2 as per user's choice:

```
<html lang="en">
<head>
    <title>Use of form</title>
    <script src='angular.min.js'></script>
</head>
<body>
    <div ng-app>
      <form>
        Disable form fields:<input type="checkbox" ng-model="option">
<br/>
        Text1:<input type="text" ng-model="fname" name="fname" ng-
disabled="option"> <br/>
        Text2:<input type="text" ng-model="sname" name="sname"ng-
disabled="option"><br/> <br/>
        Text3:<input type="text" ng-model="tname" name="tname"><br/>
<br/>
      </form>
    </div>
</body>
</html>
```

ng-click:

- The ng-click directive tells AngularJS what to do when an HTML element is clicked.
- Expression or function can be executed when respective button is clicked using ng-click directive.

Implementation of ng-click using expression:

```
<html lang="en">
<head>
    <title>Use of form</title>
    <script src='angular.min.js'></script>
</head>
<body>
    <div ng-app ng-init="cnt=0">
      <button ng-click="cnt=cnt+1">Click</button>
      <button ng-click="cnt=0">Reset</button>
      Count={{cnt}}
    </div>
</body>
</html>
```

Implementation of ng-click using function:

```
<html lang="en">
<head>
    <title>Use of form</title>
    <script src='angular.min.js'></script>
</head>
<body ng-app="myApp">
  <div ng-controller="myCtrl">
      <button ng-click="myFunc()">OK</button>
      <p>The button has been clicked {{count}} times.</p>
  </div>
  <script>
  angular.module('myApp', [])
  .controller('myCtrl', ['$scope', function($scope) {
      $scope.count = 0;
      $scope.myFunc = function() {
          $scope.count++;
      };
  }]);
  </script>
  </body>
  </html>
```

AngularJS application:

Using module, controller, routing and different directives, the application can be created using AngulaJS.

Creating application using Angular JS:

```
<html lang="en">
<head>
    <title>Use of form</title>
    <script src='angular.min.js'></script>
    <script>
        var app = angular.module("myShoppingList", []);
        app.controller("myCtrl", function($scope) {
          $scope.products = ["Milk", "Bread", "Cheese"];
          $scope.addItem = function () {
            $scope.products.push($scope.addMe);
          }
        });
        </script>
</head>
<body>
    <div ng-app="myShoppingList" ng-controller="myCtrl">
        <ul>
          <li ng-repeat="x in products">{{x}}</li>
        </ul>
        <input ng-model="addMe">
        <button ng-click="addItem()">Add</button>
      </div>
  </body>
  </html>
```