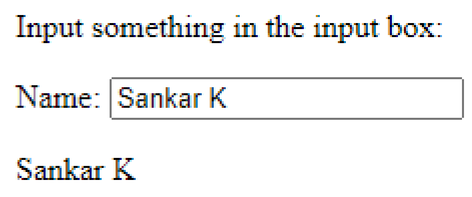
**THE NEW COLLEGE (AUTONOMOUS), CHENNAI-14.**

DEPARTMENT OF COMPUTER APPLICATIONS – SHIFT - II

**STUDY MATERIAL**

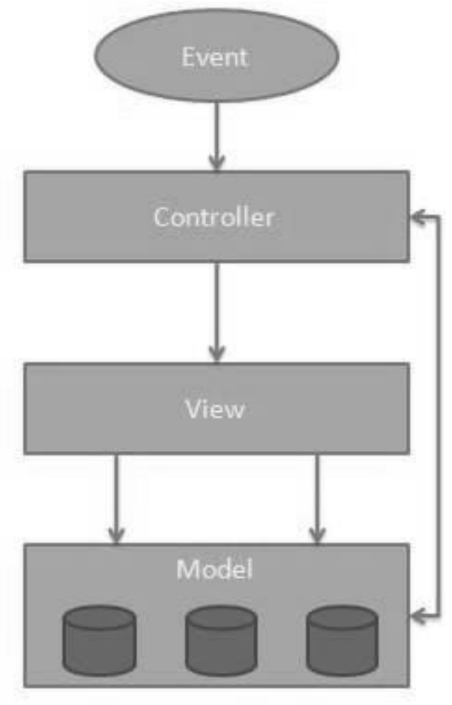
SUBJECT: OPEN SOURCE TECHNOLOGIES SUB CODE: 20BHM615

CLASS: III BCA STAFF: Dr. K. SANKAR

~~~

# UNIT-III

## Introduction to AngularJS

AngularJS extends HTML with new attributes. It is perfect for Single Page Applications (SPAs). AngularJS is a **JavaScript framework**. It can be added to an HTML page with a <script> tag. AngularJS extends HTML attributes with **Directives**, and binds data to HTML with **Expressions**.

**Example Program**

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

**Output**



```
Input something in the input box:

Name: Sankar K

Sankar K
```

## 2. ANGULARJS - MVC ARCHITECTURE

**M**odel **V**iew **C**ontroller or MVC as it is popularly called, is a software design pattern for developing web applications. A Model View Controller pattern is made up of the following three parts −
~~~

- **Model** − It is the lowest level of the pattern responsible for maintaining data.
- **View** − It is responsible for displaying all or a portion of the data to the user.
- **Controller** − It is a software Code that controls the interactions between the Model and View.

MVC is popular because it isolates the application logic from the user interface layer and supports separation of concerns. The controller receives all requests for the application and then works with the model to prepare any data needed by the view. The view then uses the data prepared by the controller to generate a final presentable response. The MVC abstraction can be graphically represented as follows.



**Model**

The model is responsible for managing application data. It responds to the request from view and to the instructions from controller to update itself.

**View**

A presentation of 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.

**Controller**

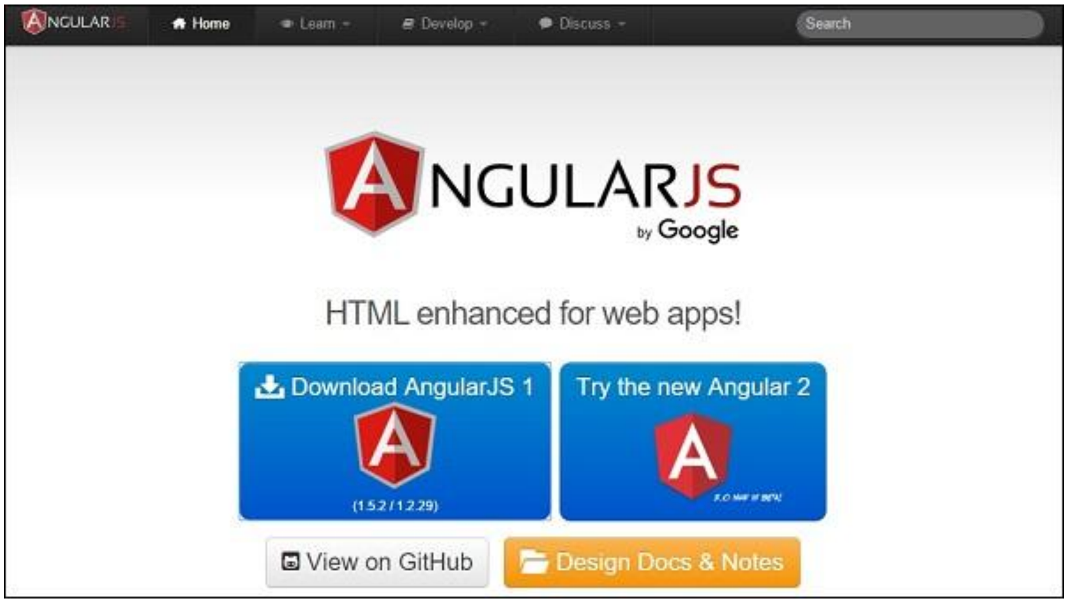
The controller responds to user input and performs interactions on the data model objects. The controller receives input, validates it, and then performs business operations that modify the state of the data model.

.

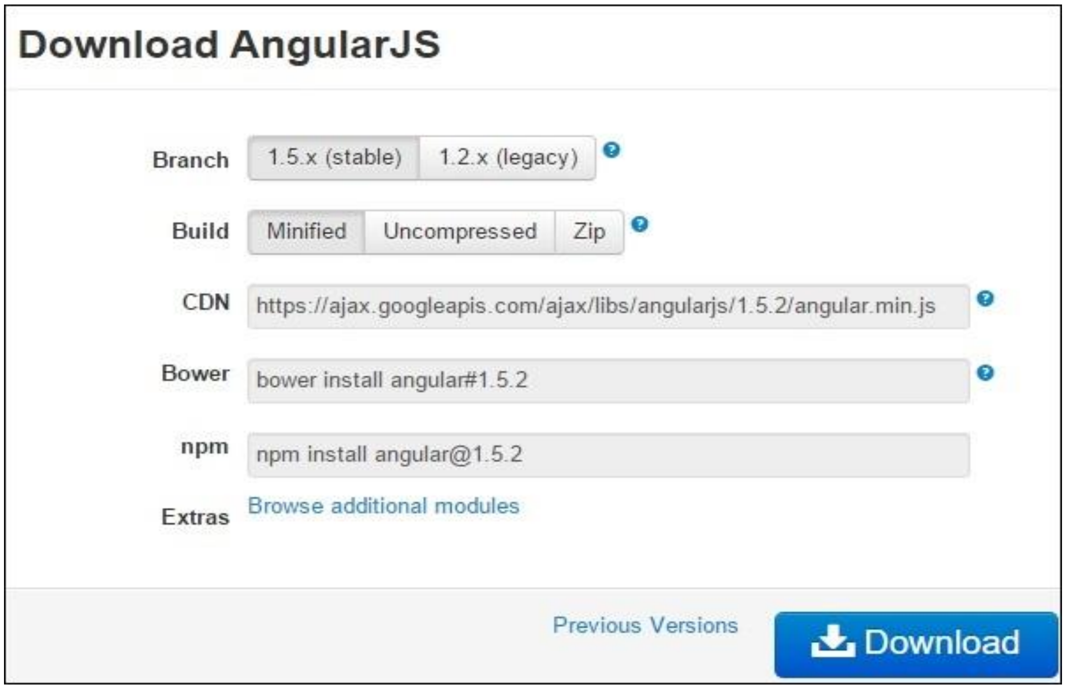
## 3. ANGULARJS - ENVIRONMENT SETUP

This chapter describes how to set up AngularJS library to be used in web application development. It also briefly describes the directory structure and its contents.

When you open the link https://angularjs.org/, you will see there are two options to download AngularJS library −



- **View on GitHub** − By clicking on this button, you are diverted to GitHub and get all the latest scripts.
- **Download AngularJS 1** − By clicking on this button, a screen you get to see a dialog box shown as −



Download AngularJS

| | |
|---|---|
| Branch | 1.5.x (stable) / 1.2.x (legacy) |
| Build | Minified / Uncompressed / Zip |
| CDN | https://ajax.googleapis.com/ajax/libs/angularjs/1.5.2/angular.min.js |
| Bower | bower install angular#1.5.2 |
| npm | npm install angular@1.5.2 |
| Extras | Browse additional modules |

This screen gives various options of using Angular JS as follows −

- **Downloading and hosting files locally**
    - There are two different options: Legacy and Latest. The names themselves are self-descriptive. The Legacy has version less than 1.2.x and the Latest come with version 1.3.x.
    - We can also go with the minimized, uncompressed, or zipped version.
- **CDN access** − You also have access to a CDN. The CDN gives you access to regional data centers. In this case, the Google host. The CDN transfers the responsibility of hosting files from your own servers to a series of external ones. It also offers an advantage that if the visitor of your web page has already downloaded a copy of AngularJS from the same CDN, there is no need to re-download it.

We are using the CDN versions of the library throughout this tutorial.

**Example**

```
<!doctype html>
<html>
  <head>
    <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.5.2/angular.min.js"></script>
  </head>

  <body ng-app = "myapp">
    <div ng-controller = "HelloController" >
      <h2>Welcome {{helloTo.title}} to the world of Tutorialspoint!</h2>
    </div>

    <script>
      angular.module("myapp", [])

      .controller("HelloController", function($scope) {
        $scope.helloTo = {};
        $scope.helloTo.title = "AngularJS";
      });
    </script>

  </body>
</html>
```

Let us go through the above code in detail −

**Include AngularJS**

We include the AngularJS JavaScript file in the HTML page so that we can use it −

```
<head>
  <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
  </script>
</head>
```

You can check the latest version of AngularJS on its official website.

**POINT TO ANGULARJS APP**

Next, it is required to tell which part of HTML contains the AngularJS app. You can do this by adding the ng-app attribute to the root HTML element of the AngularJS app. You can either add it to the html element or the body element as shown below −

```
<body ng-app = "myapp">
</body>
```

**VIEW**

```
<div ng-controller = "HelloController" >
  <h2>Welcome {{helloTo.title}} to the world of Tutorialspoint!</h2>
</div>
```

*ng-controller* tells AngularJS which controller to use with this view. *helloTo.title* tells AngularJS to write the model value named helloTo.title in HTML at this location.

**CONTROLLER**

```
<script>
  angular.module("myapp", [])
    .controller("HelloController", function($scope) {
    $scope.helloTo = {};
    $scope.helloTo.title = "AngularJS";
  });
</script>
```

This code registers a controller function named HelloController in the angular module named *myapp*. We will study more about modules and controllers in their respective chapters.

The controller function is registered in angular via the angular.module(...).controller(...) function call.

The $scope parameter model is passed to the controller function. The controller function adds a *helloTo* JavaScript object, and in that object it adds a *title* field.

**EXECUTION**

Save the above code as *myfirstexample.html* and open it in any browser. You get to see the following output −

What happens when the page is loaded in the browser? Let us see −

- HTML document is loaded into the browser, and evaluated by the browser.
- AngularJS JavaScript file is loaded, the angular *global* object is created.
- The JavaScript which registers controller functions is executed.
- Next, AngularJS scans through the HTML to search for AngularJS apps as well as views.
- Once the view is located, it connects that view to the corresponding controller function.
- Next, AngularJS executes the controller functions.
- It then renders the views with data from the model populated by the controller. The page is now ready.

## 4. ANGULARJS – DIRECTIVES

AngularJS directives are used to extend HTML. They are special attributes starting with **ng-** prefix. Let us discuss the following directives −

- **ng-app** − This directive starts an AngularJS Application.
- **ng-init** − This directive initializes application data.
- **ng-model** − This directive defines the model that is variable to be used in AngularJS.
- **ng-repeat** − This directive repeats HTML elements for each item in a collection.

### ng-app directive

The ng-app directive starts an AngularJS Application. It defines the root element. It automatically initializes or bootstraps the application when the web page containing AngularJS Application is loaded. It is also used to load various AngularJS modules in AngularJS Application.

**Example,** we define a default AngularJS application using ng-app attribute of a <div> element.

```
<div ng-app = "">
  ...
</div>
```

**ng-init directive**

The ng-init directive initializes an AngularJS Application data. It is used to assign values to the variables.

**Example**, we initialize an array of countries.

```
<div ng-app = "" ng-init = "countries = [{locale:'en-US',name:'United States'},
  {locale:'en-GB',name:'United Kingdom'}, {locale:'en-FR',name:'France'}]">
  ...
</div>
```

**ng-model directive**

The ng-model directive defines the model/variable to be used in AngularJS Application.

**Example,** we define a model named *name*.

```
<div ng-app = "">
  ...
  <p>Enter your Name: <input type = "text" ng-model = "name"></p>
</div>
```

**ng-repeat directive**

The ng-repeat directive repeats HTML elements for each item in a collection.

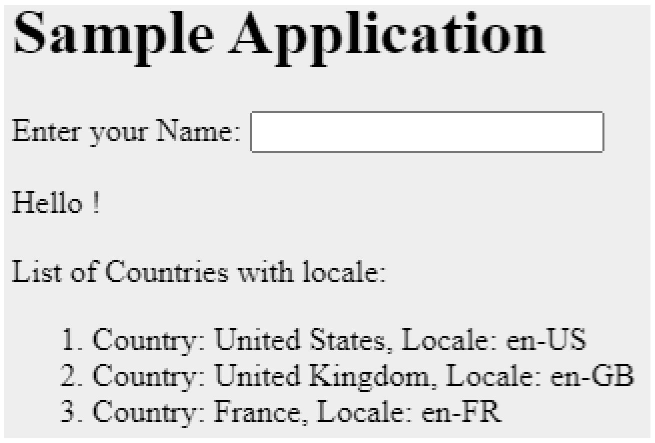
**Example**

```
<div ng-app = "">
  ...
  <p>List of Countries with locale:</p>
    <ol>
    <li ng-repeat = "country in countries">
      {{ 'Country: ' + country.name + ', Locale: ' + country.locale }}
    </li>
  </ol>
</div>
```

**Example Program**

```
<html>
  <head>
    <title>AngularJS Directives</title>
  </head>
  <body>
    <h1>Sample Application</h1>

    <div ng-app = "" ng-init = "countries = [{locale:'en-US',name:'United States'},
      {locale:'en-GB',name:'United Kingdom'}, {locale:'en-FR',name:'France'}]">
      <p>Enter your Name: <input type = "text" ng-model = "name"></p>
      <p>Hello <span ng-bind = "name"></span>!</p>
      <p>List of Countries with locale:</p>

      <ol>
        <li ng-repeat = "country in countries">
          {{ 'Country: ' + country.name + ', Locale: ' + country.locale }}
        </li>
      </ol>
    </div>

    <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
    </script>

  </body>
</html>
```

**Output**



```
Sample Application

Enter your Name:

Hello !

List of Countries with locale:

    1. Country: United States, Locale: en-US
    2. Country: United Kingdom, Locale: en-GB
    3. Country: France, Locale: en-FR
```

## 5. ANGULARJS – EXPRESSIONS

Expressions are used to bind application data to HTML. Expressions are written inside double curly braces such as in {{expression}}. Expressions behave similar to ngbind directives. AngularJS expressions are pure JavaScript expressions and output the data where they are used.

**Using numbers**

<p>Expense on Books : {{cost * quantity}} Rs</p>

**Using Strings**

<p>Hello {{student.firstname + " " + student.lastname}}!</p>

**Using Object**

<p>Roll No: {{student.rollno}}</p>

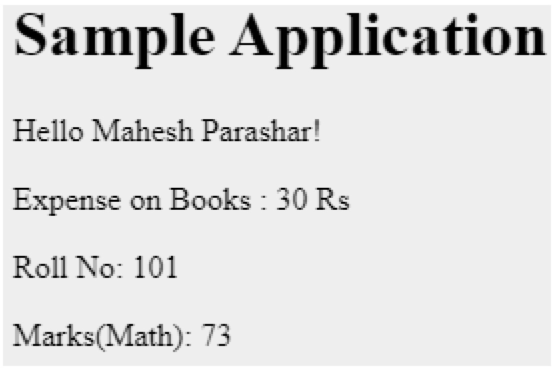
**Using Array**

<p>Marks(Math): {{marks[3]}}</p>

**Example**

```
<html>
  <head>
    <title>AngularJS Expressions</title>
  </head>

  <body>
    <h1>Sample Application</h1>

    <div ng-app = "" ng-init = "quantity = 1;cost = 30;
      student = {firstname:'Mahesh',lastname:'Parashar',rollno:101};
      marks = [80,90,75,73,60]">
      <p>Hello {{student.firstname + " " + student.lastname}}!</p>
      <p>Expense on Books : {{cost * quantity}} Rs</p>
      <p>Roll No: {{student.rollno}}</p>
      <p>Marks(Math): {{marks[3]}}</p>
    </div>

    <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
    </script>

  </body>
</html>
```

**Output**



```
Sample Application

Hello Mahesh Parashar!

Expense on Books : 30 Rs

Roll No: 101

Marks(Math): 73
```

## 6. ANGULARJS – CONTROLLERS

AngularJS application mainly relies on controllers to control the flow of data in the application. A controller is defined using *ng-controller* directive. A controller is a JavaScript object that contains attributes/properties, and functions. Each controller accepts $scope as a parameter, which refers to the application/module that the controller needs to handle.

```
<div ng-app = "" ng-controller = "studentController">
  ...
</div>
```

**Here, we declare a controller named *studentController*, using the ng-controller directive. We define it as follows −**

```
<script>
  function studentController($scope) {
    $scope.student = {
      firstName: "Mahesh",
      lastName: "Parashar",
       fullName: function() {
        var studentObject;
        studentObject = $scope.student;
        return studentObject.firstName + " " + studentObject.lastName;
      }
    };
  }
</script>
```

- The studentController is defined as a JavaScript object with $scope as an argument.
- The $scope refers to application which uses the studentController object.
- The $scope.student is a property of studentController object.
- The firstName and the lastName are two properties of $scope.student object. We pass the default values to them.
- The property fullName is the function of $scope.student object, which returns the combined name.
- In the fullName function, we get the student object and then return the combined name.

- As a note, we can also define the controller object in a separate JS file and refer that file in the HTML page.

Now we can use studentController's student property using ng-model or using expressions as follows −

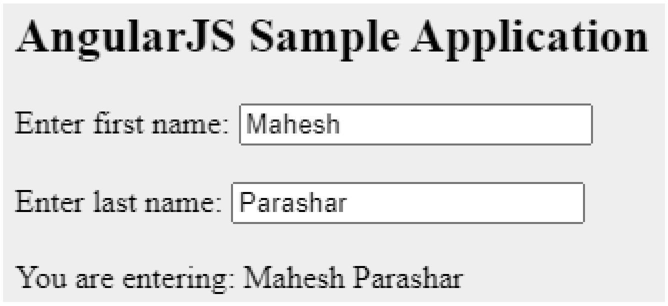
```
Enter first name: <input type = "text" ng-model = "student.firstName"><br>
Enter last name: <input type = "text" ng-model = "student.lastName"><br>
<br>
You are entering: {{student.fullName()}}
```

- We bound student.firstName and student.lastname to two input boxes.
- We bound student.fullName() to HTML.
- Now whenever you type anything in first name and last name input boxes, you can see the full name getting updated automatically.

**Example Program (**testAngularJS.htm)

```
<html>
  <head>
    <title>Angular JS Controller</title>
    <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
    </script>
  </head>

  <body>
    <h2>AngularJS Sample Application</h2>

    <div ng-app = "mainApp" ng-controller = "studentController">
      Enter first name: <input type = "text" ng-model = "student.firstName"><br>
      <br>
      Enter last name: <input type = "text" ng-model = "student.lastName"><br>
      <br>
      You are entering: {{student.fullName()}}
    </div>

    <script>
      var mainApp = angular.module("mainApp", []);

      mainApp.controller('studentController', function($scope) {
        $scope.student = {
          firstName: "Mahesh",
            lastName: "Parashar",

            fullName: function() {
               var studentObject;
               studentObject = $scope.student;
               return studentObject.firstName + " " + studentObject.lastName;
            }
         };
      });
   </script>

  </body>
</html>
```

**Output**



AngularJS Sample Application

Enter first name: Mahesh

Enter last name: Parashar

You are entering: Mahesh Parashar

**7. ANGULARJS – FILTERS**

Filters are used to modify the data. They can be clubbed in expression or directives using pipe (|) character. The following list shows the commonly used filters.

| Sr.No. | Name & Description |
|---|---|
| 1 | **uppercase**<br>converts a text to upper case text. |
| 2 | **lowercase**<br>converts a text to lower case text. |
| 3 | **currency**<br>formats text in a currency format. |
| 4 | **filter** |

| | |
|---|---|
| | filter the array to a subset of it based on provided criteria. |
| 5 | **orderby**<br>orders the array based on provided criteria. |

Uppercase Filter

Add uppercase filter to an expression using pipe character. Here we've added uppercase filter to print student name in all capital letters.

```
Enter first name:<input type = "text" ng-model = "student.firstName">
Enter last name: <input type = "text" ng-model = "student.lastName">
Name in Upper Case: {{student.fullName() | uppercase}}
```

Lowercase Filter

Add lowercase filter to an expression using pipe character. Here we've added lowercase filter to print student name in all lowercase letters.

```
Enter first name:<input type = "text" ng-model = "student.firstName">
Enter last name: <input type = "text" ng-model = "student.lastName">
Name in Lower Case: {{student.fullName() | lowercase}}
```

Currency Filter

Add currency filter to an expression returning number using pipe character. Here we've added currency filter to print fees using currency format.

```
Enter fees: <input type = "text" ng-model = "student.fees">
fees: {{student.fees | currency}}
```

Filter

To display only required subjects, we use subjectName as filter.

```
Enter subject: <input type = "text" ng-model = "subjectName">
Subject:
<ul>
  <li ng-repeat = "subject in student.subjects | filter: subjectName">
    {{ subject.name + ', marks:' + subject.marks }}
  </li>
</ul>
```

OrderBy Filter

To order subjects by marks, we use orderBy marks.

```
Subject:
<ul>
  <li ng-repeat = "subject in student.subjects | orderBy:'marks'">
    {{ subject.name + ', marks:' + subject.marks }}
  </li>
```

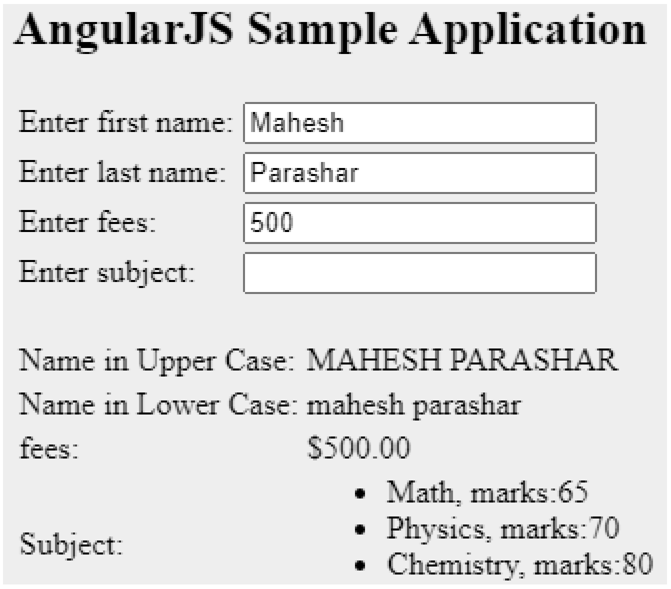
```
</ul>
```

Example

testAngularJS.htm

```
<html>
  <head>
    <title>Angular JS Filters</title>
    <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
    </script>
  </head>

  <body>
    <h2>AngularJS Sample Application</h2>

    <div ng-app = "mainApp" ng-controller = "studentController">
      <table border = "0">
        <tr>
          <td>Enter first name:</td>
          <td><input type = "text" ng-model = "student.firstName"></td>
        </tr>
        <tr>
          <td>Enter last name: </td>
          <td><input type = "text" ng-model = "student.lastName"></td>
        </tr>
        <tr>
          <td>Enter fees: </td>
          <td><input type = "text" ng-model = "student.fees"></td>
        </tr>
        <tr>
          <td>Enter subject: </td>
          <td><input type = "text" ng-model = "subjectName"></td>
        </tr>
      </table>
      <br/>

      <table border = "0">
        <tr>
          <td>Name in Upper Case: </td><td>{{student.fullName() | uppercase}}</td>
        </tr>
        <tr>
          <td>Name in Lower Case: </td><td>{{student.fullName() | lowercase}}</td>
        </tr>
        <tr>
          <td>fees: </td><td>{{student.fees | currency}}
          </td>
        </tr>
        <tr>
          <td>Subject:</td>
          <td>
            <ul>
              <li ng-repeat = "subject in student.subjects | filter: subjectName |orderBy:'marks'">
                {{ subject.name + ', marks:' + subject.marks }}
              </li>
            </ul>
          </td>
        </tr>
      </table>
    </div>

    <script>
      var mainApp = angular.module("mainApp", []);

      mainApp.controller('studentController', function($scope) {
        $scope.student = {
          firstName: "Mahesh",
          lastName: "Parashar",
          fees:500,

          subjects:[
            {name:'Physics',marks:70},
            {name:'Chemistry',marks:80},
            {name:'Math',marks:65}
          ],
          fullName: function() {
            var studentObject;
            studentObject = $scope.student;
            return studentObject.firstName + " " + studentObject.lastName;
          }
        };
      });
    </script>

  </body>
</html>
```

Output



**AngularJS Sample Application**

```
Enter first name:  Mahesh
Enter last name:   Parashar
Enter fees:        500
Enter subject:

Name in Upper Case: MAHESH PARASHAR
Name in Lower Case: mahesh parashar
fees:               $500.00
                      • Math, marks:65
Subject:              • Physics, marks:70
                      • Chemistry, marks:80
```

**8. ANGULARJS - TABLES**

Table data is generally repeatable. The ng-repeat directive can be used to draw table easily. The following example shows the use of **ng-repeat directive** to draw a table

```
<table>
  <tr>
    <th>Name</th>
    <th>Marks</th>
  </tr>

  <tr ng-repeat = "subject in student.subjects">
    <td>{{ subject.name }}</td>
    <td>{{ subject.marks }}</td>
  </tr>
</table>
```

**Table can be styled using CSS Styling.**

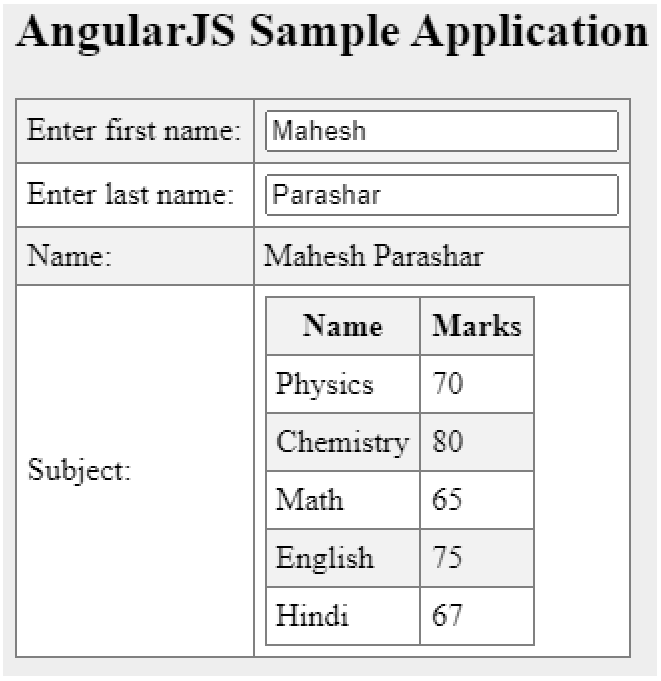
```
<style>
  table, th , td {
    border: 1px solid grey;
    border-collapse: collapse;
    padding: 5px;
  }
  table tr:nth-child(odd) {
    background-color: #f2f2f2;
  }
  table tr:nth-child(even) {
    background-color: #ffffff;
  }
</style>
```

**Example**

```
<html>
  <head>
    <title>Angular JS Table</title>
    <script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

    <style>
      table, th , td {
        border: 1px solid grey;
        border-collapse: collapse;
        padding: 5px;
      }
      table tr:nth-child(odd) {
        background-color: #f2f2f2;
      }
      table tr:nth-child(even) {
        background-color: #ffffff;
      }
    </style>
  </head>

  <body>
    <h2>AngularJS Sample Application</h2>
    <div ng-app = "mainApp" ng-controller = "studentController">

      <table border = "0">
        <tr>
          <td>Enter first name:</td>
          <td><input type = "text" ng-model = "student.firstName"></td>
        </tr>
        <tr>
          <td>Enter last name: </td>
          <td>
            <input type = "text" ng-model = "student.lastName">
          </td>
        </tr>
        <tr>
          <td>Name: </td>
          <td>{{student.fullName()}}</td>
        </tr>
        <tr>
          <td>Subject:</td>

          <td>
            <table>
              <tr>
                <th>Name</th>.
                <th>Marks</th>
```

```
               </tr>
               <tr ng-repeat = "subject in student.subjects">
                  <td>{{ subject.name }}</td>
                  <td>{{ subject.marks }}</td>
               </tr>
            </table>
         </td>
      </tr>
   </table>
</div>

<script>
   var mainApp = angular.module("mainApp", []);

   mainApp.controller('studentController', function($scope) {
      $scope.student = {
         firstName: "Mahesh",
         lastName: "Parashar",
         fees:500,

         subjects:[
            {name:'Physics',marks:70},
            {name:'Chemistry',marks:80},
            {name:'Math',marks:65},
            {name:'English',marks:75},
            {name:'Hindi',marks:67}
         ],
         fullName: function() {
            var studentObject;
            studentObject = $scope.student;
            return studentObject.firstName + " " + studentObject.lastName;
         }
      };
   });
</script>

  </body>
</html>
```

**Output**



AngularJS Sample Application

| | |
|---|---|
| Enter first name: | Mahesh |
| Enter last name: | Parashar |
| Name: | Mahesh Parashar |
| Subject: | |

| Name | Marks |
|---|---|
| Physics | 70 |
| Chemistry | 80 |
| Math | 65 |
| English | 75 |
| Hindi | 67 |

## 9. ANGULARJS - MODULES

AngularJS supports modular approach. Modules are used to separate logic such as services, controllers, application etc. from the code and maintain the code clean. We define modules in separate js files and name them as per the module.js file. In the following example, we are going to create two modules −

- **Application Module** − used to initialize an application with controller(s).
- **Controller Module** − used to define the controller.

Application Module

Here is a file named *mainApp.js* that contains the following code −

var mainApp = angular.module("mainApp", []);

Here, we declare an application **mainApp** module using angular.module function and pass an empty array to it. This array generally contains dependent modules.

Controller Module

studentController.js

```
mainApp.controller("studentController", function($scope) {
  $scope.student = {
    firstName: "Mahesh",
    lastName: "Parashar",
    fees:500,

    subjects:[
      {name:'Physics',marks:70},
      {name:'Chemistry',marks:80},
      {name:'Math',marks:65},
      {name:'English',marks:75},
      {name:'Hindi',marks:67}
    ],
    fullName: function() {
      var studentObject;
      studentObject = $scope.student;
      return studentObject.firstName + " " + studentObject.lastName;
    }
  };
});
```

Here, we declare a controller **studentController** module using mainApp.controller function.

Use Modules

```
<div ng-app = "mainApp" ng-controller = "studentController">
   ...
   <script src = "mainApp.js"></script>
   <script src = "studentController.js"></script>

</div>
```

Here, we use application module using ng-app directive, and controller using ngcontroller directive. We import the mainApp.js and studentController.js in the main HTML page.

Example (testAngularJS.htm)

```
<html>
   <head>
      <title>Angular JS Modules</title>
      <script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
      <script src = "/angularjs/src/module/mainApp.js"></script>
      <script src = "/angularjs/src/module/studentController.js"></script>

      <style>
         table, th , td {
            border: 1px solid grey;
            border-collapse: collapse;
            padding: 5px;
         }
         table tr:nth-child(odd) {
            background-color: #f2f2f2;
         }
         table tr:nth-child(even) {
            background-color: #ffffff;
         }
      </style>
   </head>

   <body>
      <h2>AngularJS Sample Application</h2>
      <div ng-app = "mainApp" ng-controller = "studentController">
         <table border = "0">
            <tr>
               <td>Enter first name:</td>
               <td><input type = "text" ng-model = "student.firstName"></td>
            </tr>
            <tr>
               <td>Enter last name: </td>
               <td><input type = "text" ng-model = "student.lastName"></td>
            </tr>
            <tr>
               <td>Name: </td>
               <td>{{student.fullName()}}</td>
            </tr>
            <tr>
               <td>Subject:</td>

               <td>
                  <table>
                     <tr>
                        <th>Name</th>
                        <th>Marks</th>
                     </tr>
                     <tr ng-repeat = "subject in student.subjects">
                        <td>{{ subject.name }}</td>
                        <td>{{ subject.marks }}</td>
                     </tr>
                  </table>
               </td>
            </tr>
         </table>
      </div>

   </body>
</html>
```

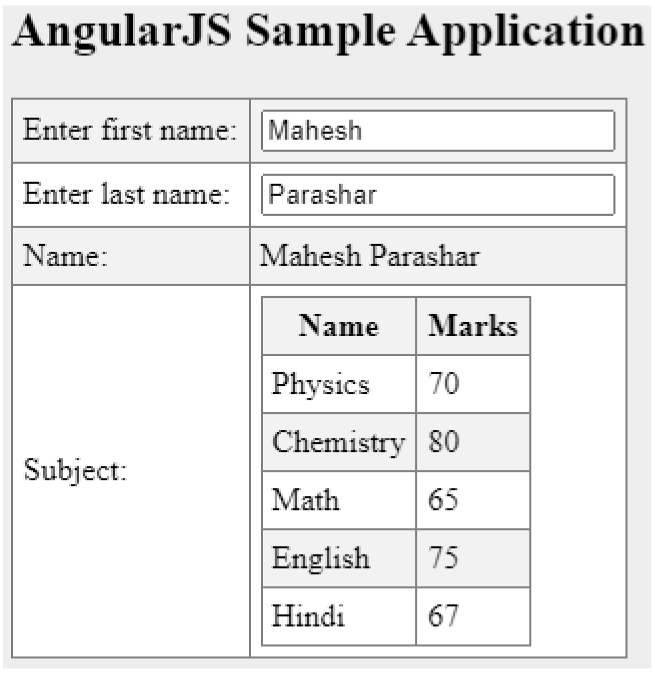
mainApp.js

```
var mainApp = angular.module("mainApp", []);
```

studentController.js

```
mainApp.controller("studentController", function($scope) {
  $scope.student = {
    firstName: "Mahesh",
    lastName: "Parashar",
    fees:500,

    subjects:[
      {name:'Physics',marks:70},
      {name:'Chemistry',marks:80},
      {name:'Math',marks:65},
      {name:'English',marks:75},
      {name:'Hindi',marks:67}
    ],
    fullName: function() {
      var studentObject;
      studentObject = $scope.student;
      return studentObject.firstName + " " + studentObject.lastName;
    }
  };
});
```

Output



AngularJS Sample Application

|  |  |
|---|---|
| Enter first name: | Mahesh |
| Enter last name: | Parashar |
| Name: | Mahesh Parashar |
| Subject: |  |

| Name | Marks |
|---|---|
| Physics | 70 |
| Chemistry | 80 |
| Math | 65 |
| English | 75 |
| Hindi | 67 |

# 10. AngularJS – Forms

AngularJS enriches form filling and validation. We can use ng-click event to handle the click button and use $dirty and $invalid flags to do the validation in a seamless way. Use novalidate with a form declaration to disable any browser-specific validation. The form controls make heavy use of AngularJS events. Let us have a look at the events first.

## Events

AngularJS provides multiple events associated with the HTML controls. For example, ng-click directive is generally associated with a button. AngularJS supports the following events –

- ng-click
- ng-dbl-click
- ng-mousedown
- ng-mouseup
- ng-mouseenter

- ng-mouseleave
- ng-mousemove
- ng-mouseover
- ng-keydown
- ng-keyup
- ng-keypress
- ng-change

ng-click

Reset data of a form using on-click directive of a button.

```
<input name = "firstname" type = "text" ng-model = "firstName" required>
<input name = "lastname" type = "text" ng-model = "lastName" required>
<input name = "email" type = "email" ng-model = "email" required>
<button ng-click = "reset()">Reset</button>

<script>
  function studentController($scope) {
    $scope.reset = function() {
      $scope.firstName = "Mahesh";
      $scope.lastName = "Parashar";
      $scope.email = "MaheshParashar@tutorialspoint.com";
    }

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

Validate Data

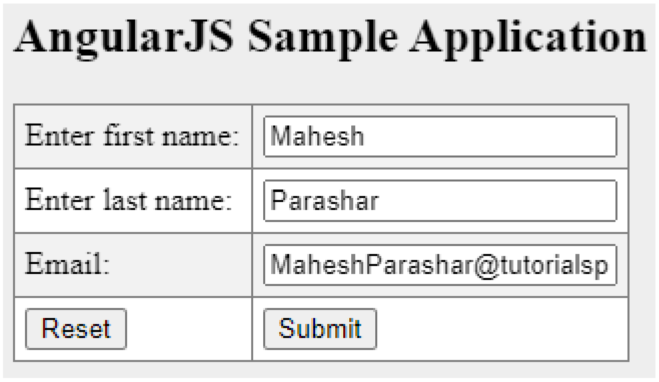
The following can be used to track error.

- **$dirty** − states that value has been changed.
- **$invalid** − states that value entered is invalid.
- **$error** − states the exact error.

Example (testAngularJS.htm)

```
<html>
  <head>
    <title>Angular JS Forms</title>
    <script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

    <style>
      table, th , td {
        border: 1px solid grey;
        border-collapse: collapse;
        padding: 5px;
      }
      table tr:nth-child(odd) {
        background-color: #f2f2f2;
      }
      table tr:nth-child(even) {
        background-color: #ffffff;
      }
    </style>

  </head>
  <body>

    <h2>AngularJS Sample Application</h2>
    <div ng-app = "mainApp" ng-controller = "studentController">

      <form name = "studentForm" novalidate>
        <table border = "0">
          <tr>
            <td>Enter first name:</td>
            <td><input name = "firstname" type = "text" ng-model = "firstName" required>
              <span style = "color:red" ng-show = "studentForm.firstname.$dirty &&
studentForm.firstname.$invalid">
                <span ng-show = "studentForm.firstname.$error.required">First Name is
required.</span>
              </span>
            </td>
          </tr>

          <tr>
            <td>Enter last name: </td>
            <td><input name = "lastname"  type = "text" ng-model = "lastName" required>
              <span style = "color:red" ng-show = "studentForm.lastname.$dirty &&
studentForm.lastname.$invalid">
                <span ng-show = "studentForm.lastname.$error.required">Last Name is
required.</span>
              </span>
            </td>
          </tr>

          <tr>
            <td>Email: </td><td><input name = "email" type = "email" ng-model = "email"
length = "100" required>
              <span style = "color:red" ng-show = "studentForm.email.$dirty &&
studentForm.email.$invalid">
                <span ng-show = "studentForm.email.$error.required">Email is required.</span>
                <span ng-show = "studentForm.email.$error.email">Invalid email
address.</span>
              </span>
            </td>
          </tr>
          <tr>
            <td>
              <button ng-click = "reset()">Reset</button>
            </td>
            <td>
              <button ng-disabled = "studentForm.firstname.$dirty &&
                studentForm.firstname.$invalid || studentForm.lastname.$dirty &&
                studentForm.lastname.$invalid || studentForm.email.$dirty &&
                studentForm.email.$invalid" ng-click="submit()">Submit</button>
            </td>
          </tr>

        </table>
      </form>
    </div>

    <script>
      var mainApp = angular.module("mainApp", []);

      mainApp.controller('studentController', function($scope) {
        $scope.reset = function() {
          $scope.firstName = "Mahesh";
          $scope.lastName = "Parashar";
          $scope.email = "MaheshParashar@tutorialspoint.com";
        }

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

  </body>
</html>
```

**Output**



## AngularJS Sample Application

|  |  |
|---|---|
| Enter first name: | Mahesh |
| Enter last name: | Parashar |
| Email: | MaheshParashar@tutorialsp |
| Reset | Submit |

## 11. ANGULARJS - VIEWS

AngularJS supports Single Page Application via multiple views on a single page. To do this, AngularJS has provided ng-view and ng-template directives, and $routeProvider services.

**ng-view Directive**

The ng-view directive simply creates a place holder where a corresponding view (HTML or ng-template view) can be placed based on the configuration.

**Usage**

Define a div with ng-view within the main module.

```
<div ng-app = "mainApp">
  ...
  <div ng-view></div>

</div>
```

**ng-template Directive**

The ng-template directive is used to create an HTML view using script tag. It contains *id* attribute which is used by $routeProvider to map a view with a controller.

**Usage**

Define a script block with type as ng-template within the main module.

```
<div ng-app = "mainApp">
  ...

  <script type = "text/ng-template" id = "addStudent.htm">
    <h2> Add Student </h2>
    {{message}}
  </script>

</div>
```

**$routeProvider Service**

The $routeProvider is a key service which sets the configuration of URLs, maps them with the corresponding HTML page or ng-template, and attaches a controller with the same.

**Usage 1**

Define a script block with type as ng-template within the main module.

```
<div ng-app = "mainApp">
  ...
  <script type = "text/ng-template" id = "addStudent.htm">
    <h2> Add Student </h2>
    {{message}}
  </script>
</div>
```

**Usage 2**

Define a script block with main module and set the routing configuration.

```
var mainApp = angular.module("mainApp", ['ngRoute']);
```

```
mainApp.config(['$routeProvider', function($routeProvider) {
  $routeProvider

  .when('/addStudent', {
    templateUrl: 'addStudent.htm', controller: 'AddStudentController'
  })
  .when('/viewStudents', {
    templateUrl: 'viewStudents.htm', controller: 'ViewStudentsController'
  })
  .otherwise ({
    redirectTo: '/addStudent'
  });

}]);
```

The following points are important to be considered in the above example −

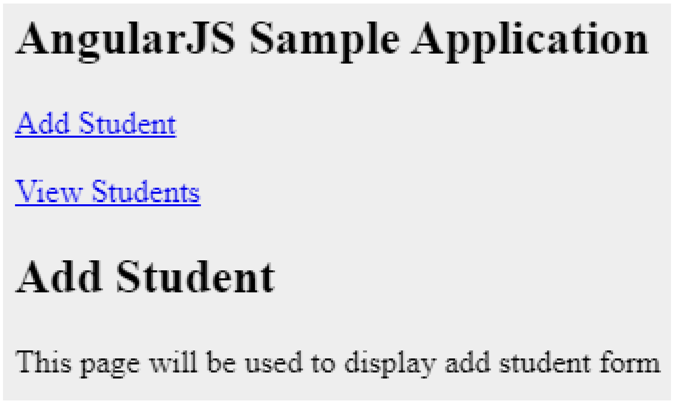
- $routeProvider is defined as a function under config of mainApp module using key as '$routeProvider'.
- $routeProvider.when defines a URL "/addStudent", which is mapped to "addStudent.htm". addStudent.htm should be present in the same path as main HTML page. If the HTML page is not defined, then ng-template needs to be used with id="addStudent.htm". We used ng-template.
- "otherwise" is used to set the default view.
- "controller" is used to set the corresponding controller for the view.

**Example (testAngularJS.htm)**

```
<html>
  <head>
    <title>Angular JS Views</title>
    <script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
    <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular-route.min.js">
    </script>
  </head>

  <body>
    <h2>AngularJS Sample Application</h2>
    <div ng-app = "mainApp">
      <p><a href = "#addStudent">Add Student</a></p>
      <p><a href = "#viewStudents">View Students</a></p>
      <div ng-view></div>

      <script type = "text/ng-template" id = "addStudent.htm">
        <h2> Add Student </h2>
        {{message}}
      </script>

      <script type = "text/ng-template" id = "viewStudents.htm">
```

```
        <h2> View Students </h2>
        {{message}}
      </script>
    </div>

    <script>
      var mainApp = angular.module("mainApp", ['ngRoute']);
      mainApp.config(['$routeProvider', function($routeProvider) {
        $routeProvider

        .when('/addStudent', {
          templateUrl: 'addStudent.htm',
          controller: 'AddStudentController'
        })
        .when('/viewStudents', {
          templateUrl: 'viewStudents.htm',
          controller: 'ViewStudentsController'
        })
        .otherwise({
          redirectTo: '/addStudent'
        });
      }]);
      mainApp.controller('AddStudentController', function($scope) {
        $scope.message = "This page will be used to display add student form";
      });
      mainApp.controller('ViewStudentsController', function($scope) {
        $scope.message = "This page will be used to display all the students";
      });
    </script>

  </body>
</html>
```

**Output**



AngularJS Sample Application

Add Student

View Students

Add Student

This page will be used to display add student form

**12. ANGULARJS – SCOPES**

Scope is a special JavaScript object that connects controller with views. Scope contains model data. In controllers, model data is accessed via $scope object.

```
<script>
  var mainApp = angular.module("mainApp", []);

  mainApp.controller("shapeController", function($scope) {
    $scope.message = "In shape controller";
    $scope.type = "Shape";
  });
</script>
```

The following important points are considered in above example −

- The $scope is passed as first argument to controller during its constructor definition.
- The $scope.message and $scope.type are the models which are used in the HTML page.
- We assign values to models that are reflected in the application module, whose controller is shapeController.
- We can define functions in $scope.

**Scope Inheritance**

Scope is controller-specific. If we define nested controllers, then the child controller inherits the scope of its parent controller.

```
<script>
  var mainApp = angular.module("mainApp", []);

  mainApp.controller("shapeController", function($scope) {
    $scope.message = "In shape controller";
    $scope.type = "Shape";
  });
  mainApp.controller("circleController", function($scope) {
    $scope.message = "In circle controller";
  });

</script>
```

The following important points are considered in above example −

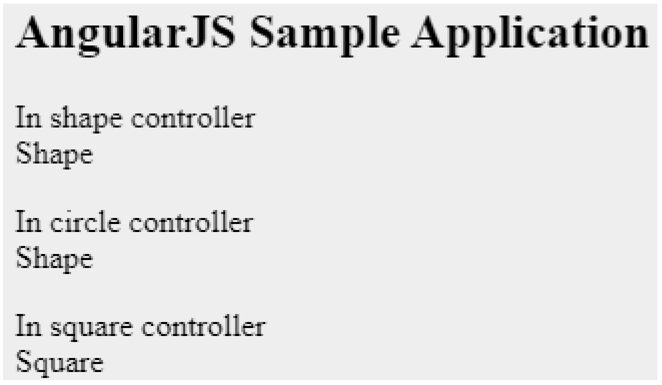
- We assign values to the models in shapeController.
- We override message in child controller named *circleController*. When message is used within the module of controller named *circleController*, the overridden message is used.

**Example (testAngularJS.htm)**

```
<html>
  <head>
    <title>Angular JS Forms</title>
  </head>

  <body>
    <h2>AngularJS Sample Application</h2>

    <div ng-app = "mainApp" ng-controller = "shapeController">
```

```
      <p>{{message}} <br/> {{type}} </p>

      <div ng-controller = "circleController">
        <p>{{message}} <br/> {{type}} </p>
      </div>

      <div ng-controller = "squareController">
        <p>{{message}} <br/> {{type}} </p>
      </div>

    </div>
    <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
    </script>

    <script>
      var mainApp = angular.module("mainApp", []);

      mainApp.controller("shapeController", function($scope) {
        $scope.message = "In shape controller";
        $scope.type = "Shape";
      });
      mainApp.controller("circleController", function($scope) {
        $scope.message = "In circle controller";
      });
      mainApp.controller("squareController", function($scope) {
        $scope.message = "In square controller";
        $scope.type = "Square";
      });

    </script>

  </body>
</html>
```

**Output**



```
AngularJS Sample Application

In shape controller
Shape

In circle controller
Shape

In square controller
Square
```

# 13. ANGULARJS – SERVICES

AngularJS supports the concept of Separation of Concerns using services architecture. Services are JavaScript functions, which are responsible to perform only specific tasks. This makes them individual entities which are maintainable and testable. The controllers and filters can call them on requirement basis. Services are normally injected using the dependency injection mechanism of AngularJS.

AngularJS provides many inbuilt services. For example, $http, $route, $window, $location, etc. Each service is responsible for a specific task such as the $http is used to make ajax call to get the server data, the $route is used to define the routing information, and so on. The inbuilt services are always prefixed with $ symbol.

There are two ways to create a service −

- Factory
- Service

## Using Factory Method

In this method, we first define a factory and then assign method to it.

```
var mainApp = angular.module("mainApp", []);
mainApp.factory('MathService', function() {
  var factory = {};

  factory.multiply = function(a, b) {
    return a * b
  }

  return factory;
});
```

## Using Service Method

In this method, we define a service and then assign method to it. We also inject an already available service to it.

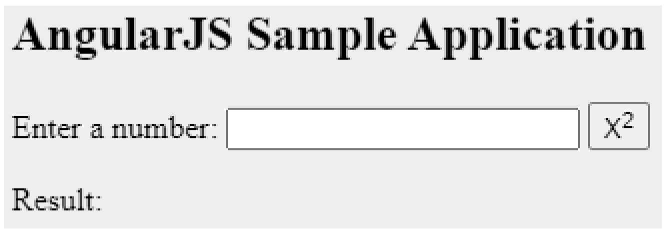
```
mainApp.service('CalcService', function(MathService) {
  this.square = function(a) {
      return MathService.multiply(a,a);
   }
});
```

**Example**

testAngularJS.htm

```
<html>
   <head>
      <title>Angular JS Services</title>
      <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
      </script>
   </head>

   <body>
      <h2>AngularJS Sample Application</h2>

      <div ng-app = "mainApp" ng-controller = "CalcController">
         <p>Enter a number: <input type = "number" ng-model = "number" /></p>
         <button ng-click = "square()">X<sup>2</sup></button>
         <p>Result: {{result}}</p>
      </div>

      <script>
         var mainApp = angular.module("mainApp", []);

         mainApp.factory('MathService', function() {
            var factory = {};

            factory.multiply = function(a, b) {
               return a * b
            }
            return factory;
         });
         mainApp.service('CalcService', function(MathService) {
            this.square = function(a) {
               return MathService.multiply(a,a);
            }
         });
         mainApp.controller('CalcController', function($scope, CalcService) {
            $scope.square = function() {
               $scope.result = CalcService.square($scope.number);
            }
         });
      </script>

   </body>
</html>
```

**Output**



**14. ANGULARJS - DEPENDENCY INJECTION**

Dependency Injection is a software design in which components are given their dependencies instead of hard coding them within the component. It relieves a component from locating the dependency and makes dependencies configurable. It also helps in making components reusable, maintainable and testable.

AngularJS provides a supreme Dependency Injection mechanism. It provides following core components which can be injected into each other as dependencies.

- Value
- Factory
- Service
- Provider
- Constant

**Value**

Value is a simple JavaScript object, which is required to pass values to the controller during config phase (config phase is when AngularJS bootstraps itself).

```
//define a module
var mainApp = angular.module("mainApp", []);

//create a value object as "defaultInput" and pass it a data.
mainApp.value("defaultInput", 5);
...

//inject the value in the controller using its name "defaultInput"
mainApp.controller('CalcController', function($scope, CalcService, defaultInput) {
   $scope.number = defaultInput;
   $scope.result = CalcService.square($scope.number);

   $scope.square = function() {
      $scope.result = CalcService.square($scope.number);
   }
});
```

**Factory**

Factory is a function which is used to return value. It creates a value on demand whenever a service or a controller requires it. It generally uses a factory function to calculate and return the value.

```
//define a module
var mainApp = angular.module("mainApp", []);

//create a factory "MathService" which provides a method multiply to return multiplication of
two numbers
mainApp.factory('MathService', function() {
   var factory = {};

   factory.multiply = function(a, b) {
      return a * b
   }
   return factory;
});

//inject the factory "MathService" in a service to utilize the multiply method of factory.
mainApp.service('CalcService', function(MathService) {
   this.square = function(a) {
      return MathService.multiply(a,a);
   }
});
...
```

**Service**

Service is a singleton JavaScript object containing a set of functions to perform certain tasks. Service is defined using service() function and it is then injected into the controllers.

```
//define a module
var mainApp = angular.module("mainApp", []);
...

//create a service which defines a method square to return square of a number.
mainApp.service('CalcService', function(MathService) {
   this.square = function(a) {
      return MathService.multiply(a,a);
   }
});

//inject the service "CalcService" into the controller
mainApp.controller('CalcController', function($scope, CalcService, defaultInput) {
   $scope.number = defaultInput;
   $scope.result = CalcService.square($scope.number);
```

```
  $scope.square = function() {
     $scope.result = CalcService.square($scope.number);
  }
});
```

**Provider**

Provider is used by AngularJS internally to create services, factory, etc. during the config phase. The following script can be used to create MathService that we created earlier. Provider is a special factory method with get() method which is used to return the value/service/factory.

```
//define a module
var mainApp = angular.module("mainApp", []);
...

//create a service using provider which defines a method square to return square of a number.
mainApp.config(function($provide) {
  $provide.provider('MathService', function() {
     this.$get = function() {
        var factory = {};

        factory.multiply = function(a, b) {
           return a * b;
        }
        return factory;
     };
  });
});
```

**Constant**

Constants are used to pass values at the config phase considering the fact that value cannot be used during the config phase.

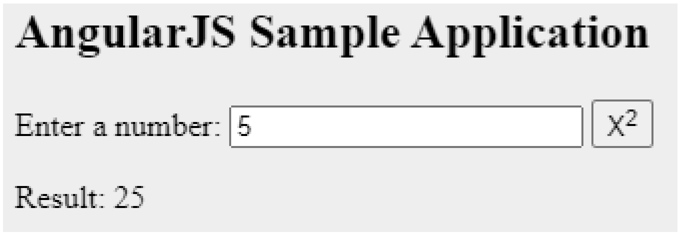
mainApp.constant("configParam", "constant value");

**Example**

testAngularJS.htm

```
<html>
  <head>
    <title>AngularJS Dependency Injection</title>
  </head>

  <body>
    <h2>AngularJS Sample Application</h2>

    <div ng-app = "mainApp" ng-controller = "CalcController">
      <p>Enter a number: <input type = "number" ng-model = "number" /></p>
      <button ng-click = "square()">X<sup>2</sup></button>
      <p>Result: {{result}}</p>
    </div>

    <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
    </script>

    <script>
      var mainApp = angular.module("mainApp", []);

      mainApp.config(function($provide) {
        $provide.provider('MathService', function() {
          this.$get = function() {
            var factory = {};

            factory.multiply = function(a, b) {
              return a * b;
            }
            return factory;
          };
        });
      });

      mainApp.value("defaultInput", 5);

      mainApp.factory('MathService', function() {
        var factory = {};

        factory.multiply = function(a, b) {
          return a * b;
        }
        return factory;
      });
      mainApp.service('CalcService', function(MathService) {
        this.square = function(a) {
          return MathService.multiply(a,a);
        }
      });
      mainApp.controller('CalcController', function($scope, CalcService, defaultInput) {
        $scope.number = defaultInput;
        $scope.result = CalcService.square($scope.number);

        $scope.square = function() {
          $scope.result = CalcService.square($scope.number);
        }
      });
    </script>

  </body>
</html>
```

**Output**


AngularJS Sample Application

Enter a number: 5 X²

Result: 25

## 15. ANGULARJS - CUSTOM DIRECTIVES

Custom directives are used in AngularJS to extend the functionality of HTML. Custom directives are defined using "directive" function. A custom directive simply replaces the element for which it is activated. AngularJS application during bootstrap finds the matching elements and do one time activity using its compile() method of the custom directive then process the element using link() method of the custom directive based on the scope of the directive. AngularJS provides support to create custom directives for following type of elements.

- **Element directives** − Directive activates when a matching element is encountered.
- **Attribute** − Directive activates when a matching attribute is encountered.
- **CSS** − Directive activates when a matching css style is encountered.
- **Comment** − Directive activates when a matching comment is encountered.

### Understanding Custom Directive

Define custom html tags.

```
<student name = "Mahesh"></student><br/>
<student name = "Piyush"></student>
```

Define custom directive to handle above custom html tags.

```
var mainApp = angular.module("mainApp", []);


//Create a directive, first parameter is the html element to be attached.
//We are attaching student html tag.
//This directive will be activated as soon as any student element is encountered in html


mainApp.directive('student', function() {
  //define the directive object
  var directive = {};

  //restrict = E, signifies that directive is Element directive
  directive.restrict = 'E';

  //template replaces the complete element with its text.
  directive.template = "Student: <b>{{student.name}}</b> ,
    Roll No: <b>{{student.rollno}}</b>";

  //scope is used to distinguish each student element based on criteria.
  directive.scope = {
    student : "=name"
  }

  //compile is called during application initialization. AngularJS calls
    it once when html page is loaded.

  directive.compile = function(element, attributes) {
    element.css("border", "1px solid #cccccc");

    //linkFunction is linked with each element with scope to get the element specific data.
    var linkFunction = function($scope, element, attributes) {
      element.html("Student: <b>"+$scope.student.name +"</b> ,
        Roll No: <b>"+$scope.student.rollno+"</b><br/>");
      element.css("background-color", "#ff00ff");
    }
    return linkFunction;
  }

  return directive;
});
```

Define controller to update the scope for directive. Here we are using name attribute's value as scope's child.

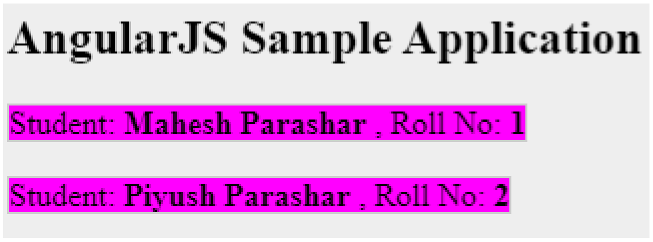
```
mainApp.controller('StudentController', function($scope) {
  $scope.Mahesh = {};
  $scope.Mahesh.name = "Mahesh Parashar";
  $scope.Mahesh.rollno  = 1;

  $scope.Piyush = {};
  $scope.Piyush.name = "Piyush Parashar";
  $scope.Piyush.rollno  = 2;
});
```

**Example**

```
<html>
  <head>
    <title>Angular JS Custom Directives</title>
  </head>

  <body>
    <h2>AngularJS Sample Application</h2>

    <div ng-app = "mainApp" ng-controller = "StudentController">
      <student name = "Mahesh"></student><br/>
      <student name = "Piyush"></student>
    </div>

    <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
    </script>

    <script>
      var mainApp = angular.module("mainApp", []);

      mainApp.directive('student', function() {
        var directive = {};
        directive.restrict = 'E';
```

```
         directive.template = "Student: <b>{{student.name}}</b> ,
           Roll No: <b>{{student.rollno}}</b>";

         directive.scope = {
           student : "=name"
         }
         directive.compile = function(element, attributes) {
           element.css("border", "1px solid #cccccc");

           var linkFunction = function($scope, element, attributes) {
             element.html("Student: <b>"+$scope.student.name +"</b> ,
               Roll No: <b>"+$scope.student.rollno+"</b><br/>");
             element.css("background-color", "#ff00ff");
           }
           return linkFunction;
         }

         return directive;
       });
       mainApp.controller('StudentController', function($scope) {
         $scope.Mahesh = {};
         $scope.Mahesh.name = "Mahesh Parashar";
         $scope.Mahesh.rollno  = 1;

         $scope.Piyush = {};
         $scope.Piyush.name = "Piyush Parashar";
         $scope.Piyush.rollno  = 2;
       });
     </script>

  </body>
</html>
```

**Output**



```
AngularJS Sample Application

Student: Mahesh Parashar , Roll No: 1

Student: Piyush Parashar , Roll No: 2
```

~~~ End of Unit-III~~~
~~~