1 MODULE

AngularJS is an open source, JavaScript based web application development framework.

AngularJS is a structural framework for dynamic web applications. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application components clearly and succinctly. Its data binding and dependency injection eliminate much of the code you currently have to write. And it all happens within the browser, making it an ideal partner with any server technology.

It was originally developed in 2009 by Misko Hevery and Adam Abrons. It is now maintained by Google.

General Features

The most important general features of AngularJS are:

- AngularJS is a efficient framework that can create Rich Internet Applications (RIA).
- AngularJS provides developers an options to write client side applications using JavaScript in a clean Model View Controller (MVC) way.
- Applications written in AngularJS are cross-browser compliant. AngularJS automatically handles JavaScript code suitable for each browser.
- AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache license version 2.0.

Overall, AngularJS is a framework to build large scale, high performance, and easy-to-maintain web applications.

Core Features

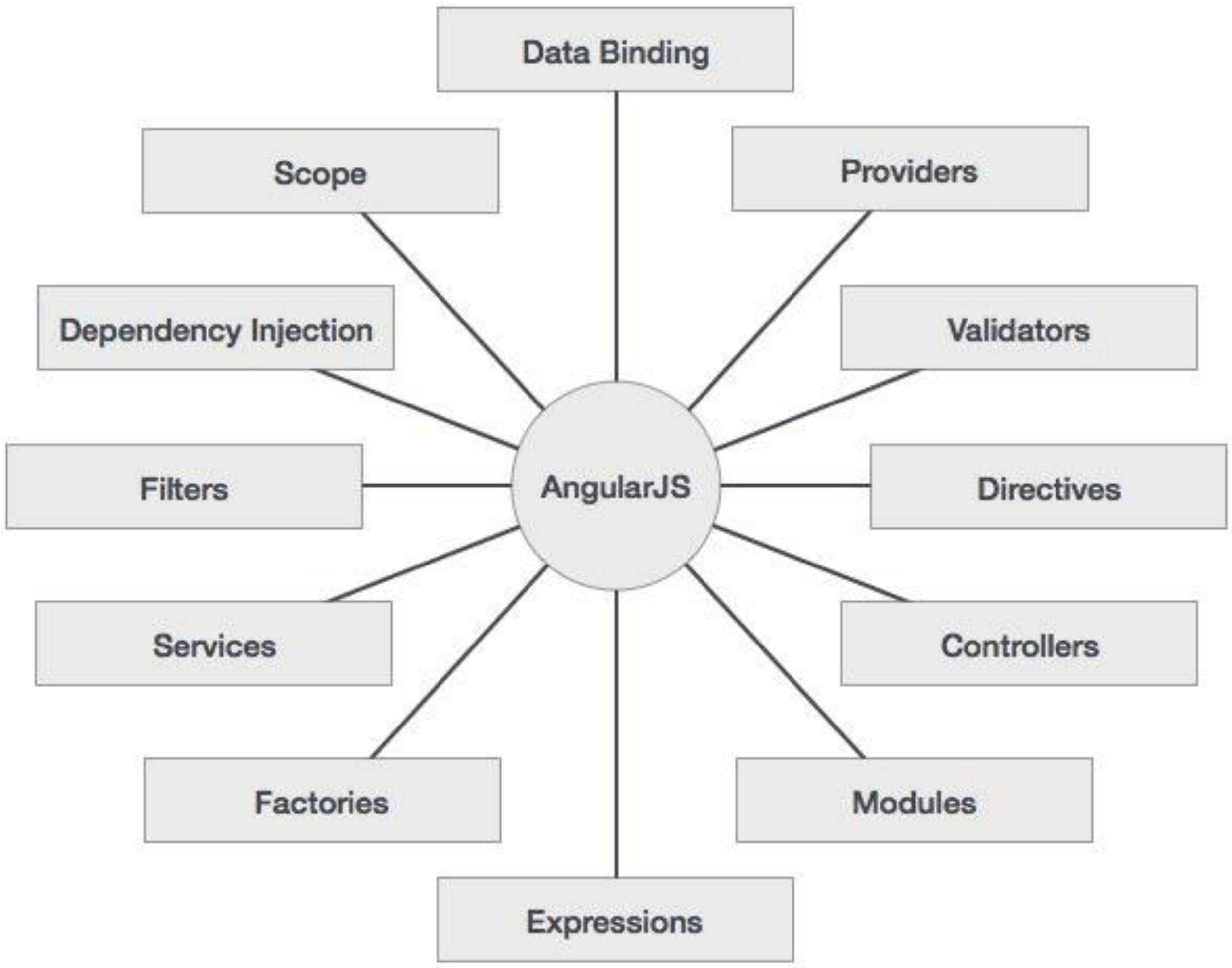
The most important core features of AngularJS are:

- **Data-binding:** It is the automatic synchronization of data between model and view components.
- **Scope:** These are objects that refer to the model. They act as a glue between controller and view.

- **Controller:** These are JavaScript functions bound to a particular scope.
- **Services:** AngularJS comes with several built-in services such as $http to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app.
- **Filters:** These select a subset of items from an array and returns a new array.
- **Directives:** Directives are markers on DOM elements such as elements, attributes, css, and more. These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives such as ngBind, ngModel etc.
- **Templates:**These are the rendered view with information from the controller and model. These can be a single file (such as index.html) or multiple views in one page using *partials*.
- **Routing:** It is concept of switching views.
- **Model View Whatever:** MVW is a design pattern for dividing an application into different parts called Model, View, and Controller, each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel). The Angular JS team refers it humorously as Model View Whatever.
- **Deep Linking:** Deep linking allows you to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state.
- **Dependency Injection:** AngularJS has a built-in dependency injection subsystem that helps the developer to create,understand, and test the applications easily.

Concepts

The following diagram depicts some important parts of AngularJS which we will discuss in detail in the subsequent chapters.



Advantages of AngularJS

The advantages of AngularJS are:

- AngularJS provides capability to create Single Page Application in a very clean and maintainable way.
- AngularJS provides data binding capability to HTML. Thus, it gives user a rich and responsive experience.
- AngularJS code is unit testable.
- AngularJS uses dependency injection and make use of separation of concerns.
- AngularJS provides reusable components.
- With AngularJS, the developers can achieve more functionality with short code.
- In AngularJS, views are pure html pages, and controllers written in JavaScript do the business processing.

On the top of everything, AngularJS applications can run on all major browsers and smart phones, including Android and iOS based phones/tablets.

Disadvantages of AngulaJS

Though AngularJS comes with a lot of merits, here are some points of concern:

- **Not Secure** : Being JavaScript only framework, application written in AngularJS are not safe. Server side authentication and authorization is must to keep an application secure.
- **Not degradable**: If the user of your application disables JavaScript, then nothing would be visible, except the basic page.

AngularJS Directives

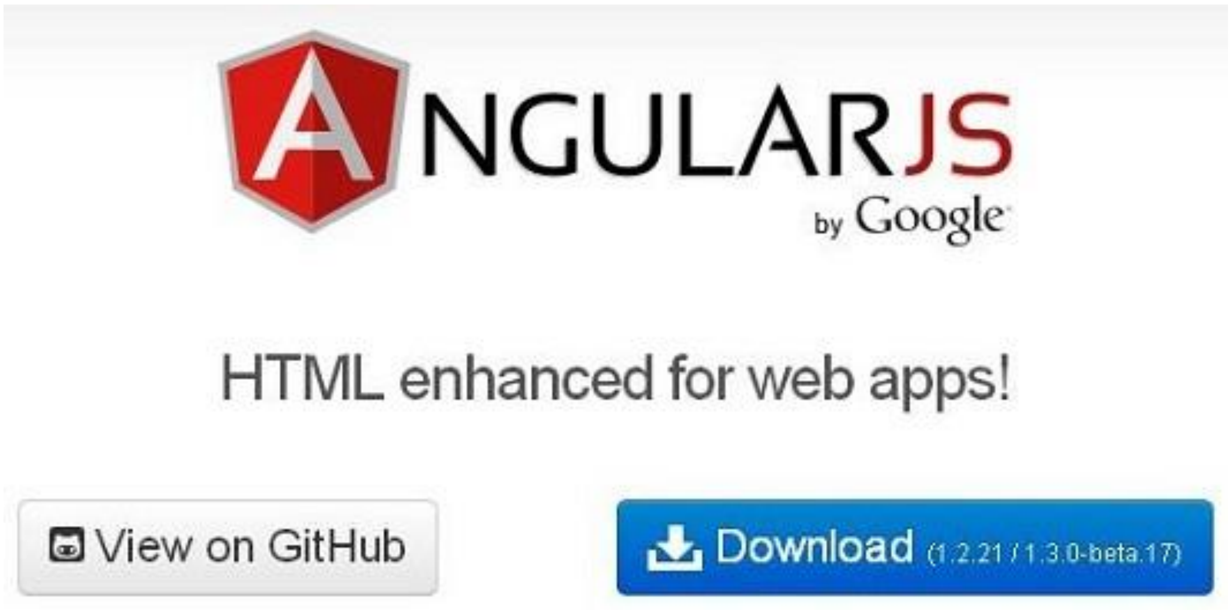
The AngularJS framework can be divided into three major parts:

- **ng-app** : This directive defines and links an AngularJS application to HTML.
- **ng-model** : This directive binds the values of AngularJS application data to HTML input controls.
- **ng-bind** : This directive binds the AngularJS application data to HTML tags.

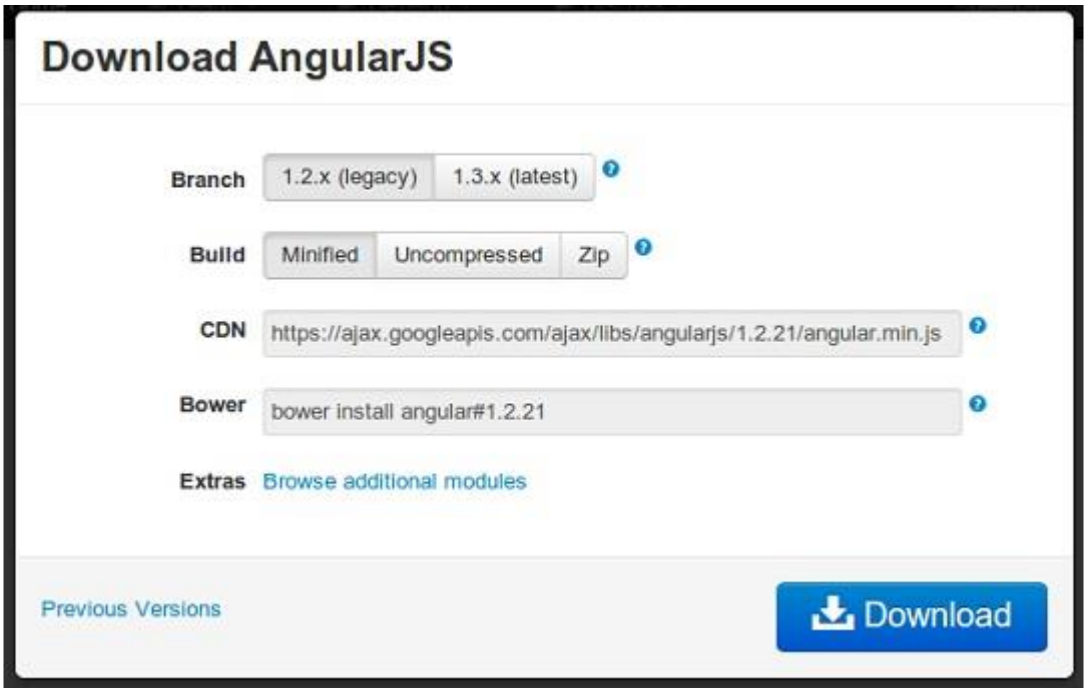
1(a). ENVIRONMENT SET UP

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:


ANGULARJS
by Google
HTML enhanced for web apps!
View on GitHub
Download (1.2.21 / 1.3.0-beta.17)

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


Download AngularJS

Branch: 1.2.x (legacy) | 1.3.x (latest)

Build: Minified | Uncompressed | Zip

CDN: https://ajax.googleapis.com/ajax/libs/angularjs/1.2.21/angular.min.js

Bower: bower install angular#1.2.21

Extras: Browse additional modules

This screen offers various options for selecting 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 around the world to regional data centres. In this case, the Google host. This means, using CDN transfers the responsibility of hosting files from your own servers to a series of external ones. This 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

Now let us write a simple example using AngularJS library. Let us create an HTML file*myfirstexample.html* shown as below:

```
<!doctype html>

<html>
```

```
   <head>

      <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.0-
beta.17/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="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.15/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 *html* element or *body* element as shown below:

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

</body>
```

View

The view is this part:

```
<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

The controller part is:

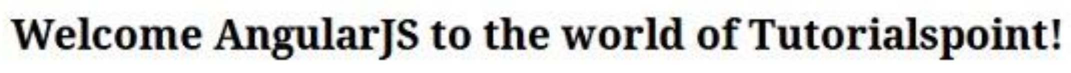
```
<script>

   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:



```
Welcome AngularJS to the world of Tutorialspoint!
```

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

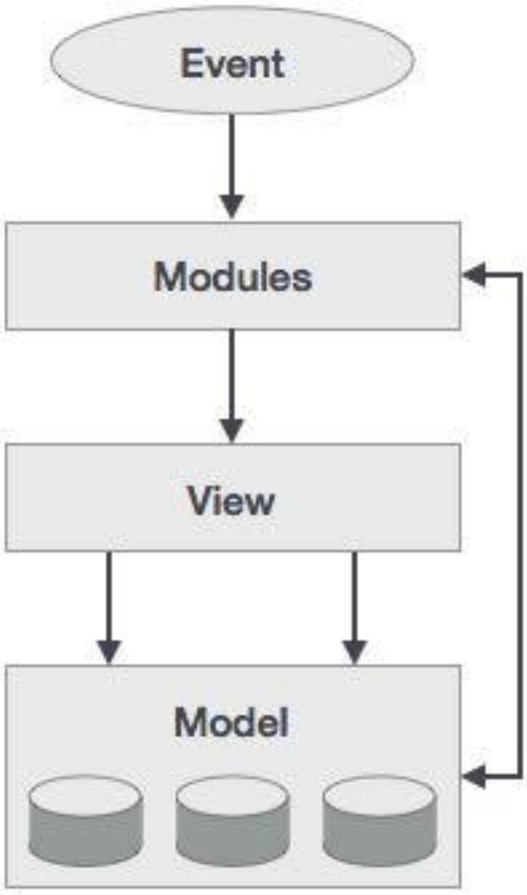
- HTML document is loaded into the browser, and evaluated by the browser.
- AngularJS JavaScript file is loaded, the angular *global* object is created.
- The JavaScript which registers controller functions is executed.
- Next, AngularJS scans through the HTML to search for AngularJS apps as well as 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.

1(b). MVC ARCHITECTURE

Model **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 as 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.



The 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.

The 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.

The 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.

AngularJS is a MVC based framework. In the coming chapters, let us see how AngularJS uses MVC methodology.

1(c). FIRST APPLICATION

Before creating actual *Hello World !* application using AngularJS, let us see the parts of a AngularJS application. An AngularJS application consists of following three important parts:

- **ng-app** : This directive defines and links an AngularJS application to HTML.
- **ng-model** : This directive binds the values of AngularJS application data to HTML input controls.
- **ng-bind** : This directive binds the AngularJS Application data to HTML tags.

Creating AngularJS Application

Step 1: Load framework

Being a pure JavaScript framework, it can be added using <Script> tag.

```
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.15/angular.min.js">

</script>
```

Step 2: Define AngularJS application using *ng-app* directive.

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

...

</div>
```

Step 3: Define a model name using *ng-model* directive.

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

Step 4: Bind the value of above model defined using ng-bind directive.

```
<p>Hello <span ng-bind="name"></span>!</p>
```

Executing AngularJS Application

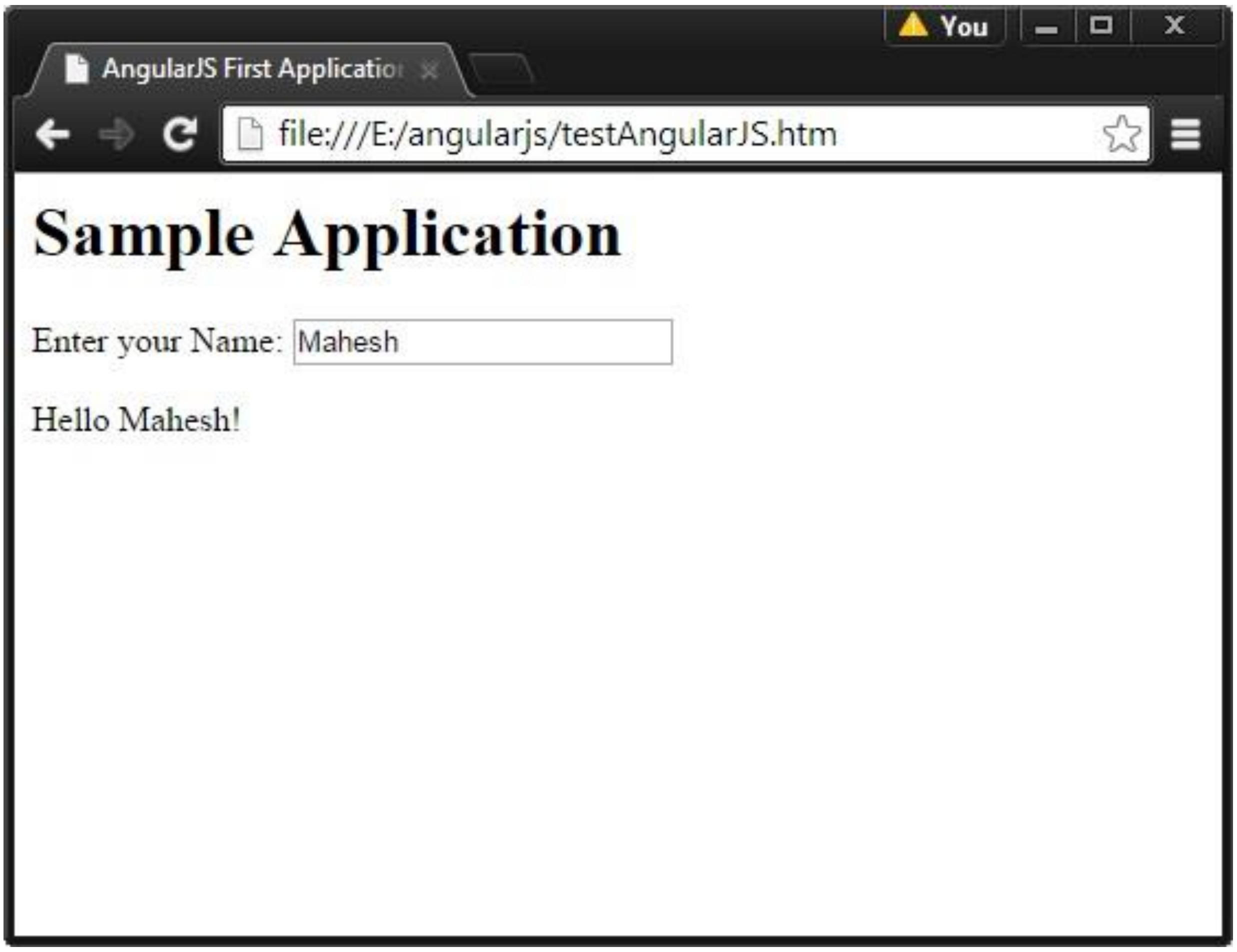
Use the above mentioned three steps in an HTML page.

testAngularJS.htm

```
<html>

<title>AngularJS First Application</title>

<body>

<h1>Sample Application</h1>

<div ng-app="">

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

   <p>Hello <span ng-bind="name"></span>!</p>

</div>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.15/angular.min.js">
</script>

</body>

</html>
```

Output

Open the file *testAngularJS.htm* in a web browser. Enter your name and see the result.


AngularJS First Applicatio
file:///E:/angularjs/testAngularJS.htm

Sample Application

Enter your Name: Mahesh

Hello Mahesh!

How AngularJS integrates with HTML

- The ng-app directive indicates the start of AngularJS application.
- The ng-model directive creates a model variable named *name*, which can be used with the HTML page and within the div having ng-app directive.
- The ng-bind then uses the *name* model to be displayed in the HTML <span> tag whenever user enters input in the text box.
- Closing </div> tag indicates the end of AngularJS application.

1(d). 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. In the following 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. In the following example, we initialize an array of countries. We use JSON syntax to define the 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. In the following 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

ng-repeat directive repeats HTML elements for each item in a collection. In the following example, we iterate over the array of countries.

```
<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

The following example shows the use of all the above mentioned directives.

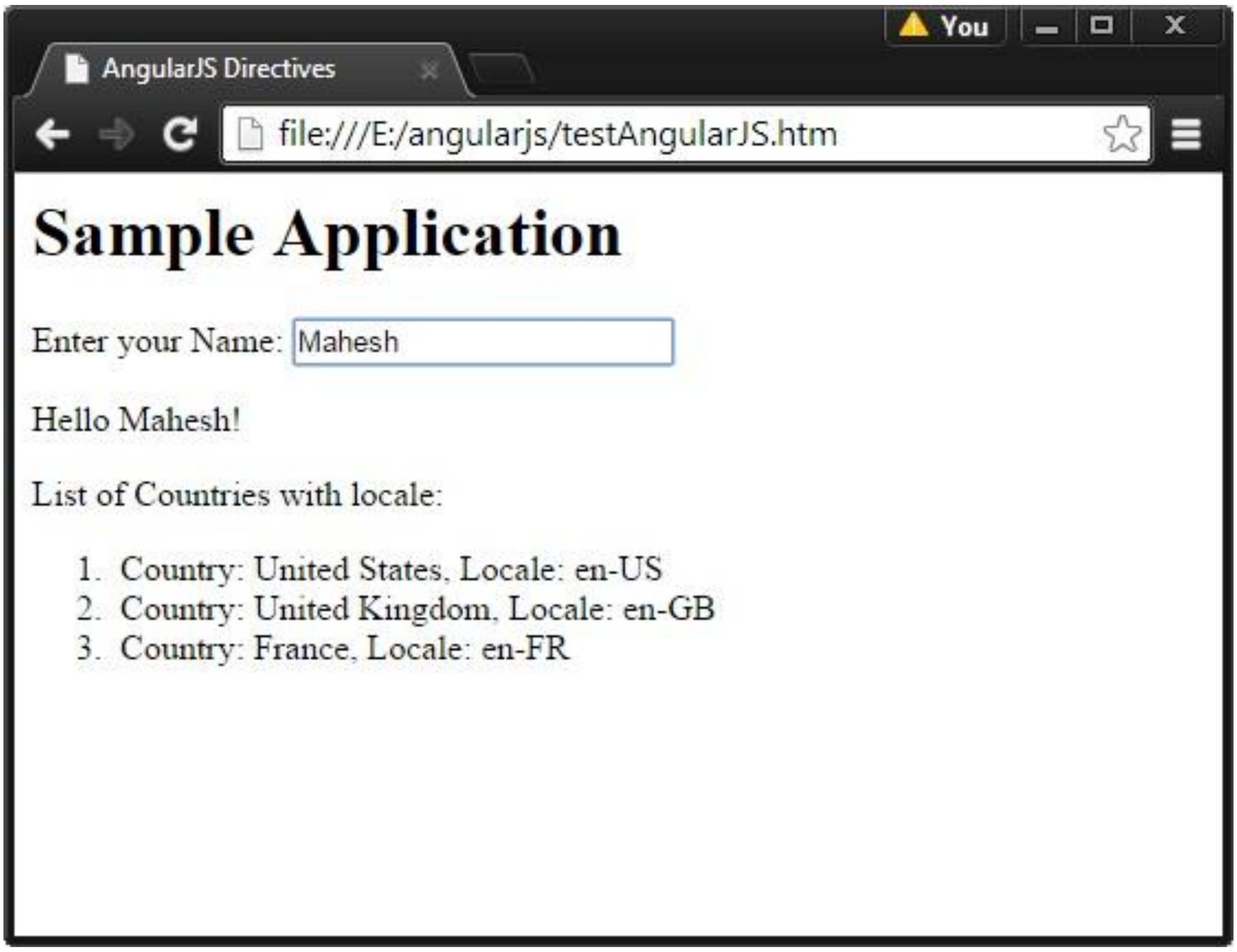
testAngularJS.htm

```
<html>

<title>AngularJS Directives</title>
```

```
<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="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.15/angular.min.js">
</script>

</body>

</html>
```

Output

Open the file *testAngularJS.htm* in a web browser. Enter your name and see the result.



Sample Application

Enter your Name: Mahesh

Hello Mahesh!

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

1(e). 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 ng-bind 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 String

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

Using Object

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

Using Array

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

Example

The following example shows use of all the above mentioned expressions:

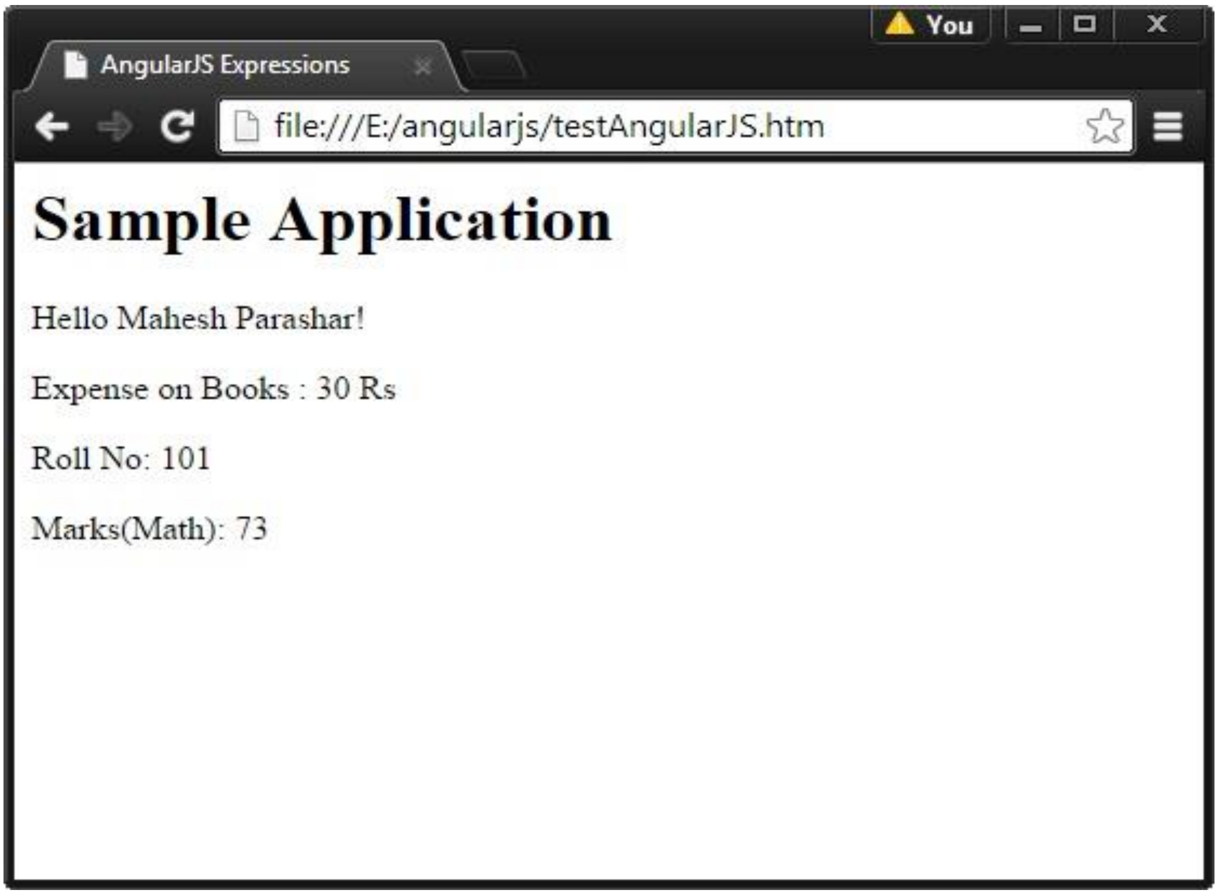
testAngularJS.htm

```
<html>

<title>AngularJS Expressions</title>

<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="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.15/angular.min.js">
</script>

</body>

</html>
```

Output

Open the file *testAngularJS.htm* in a web browser and see the result.



```
Sample Application

Hello Mahesh Parashar!

Expense on Books : 30 Rs

Roll No: 101

Marks(Math): 73
```

1(f). 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 ng-controller directive. As a next step, 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

The following example shows the use of controller:

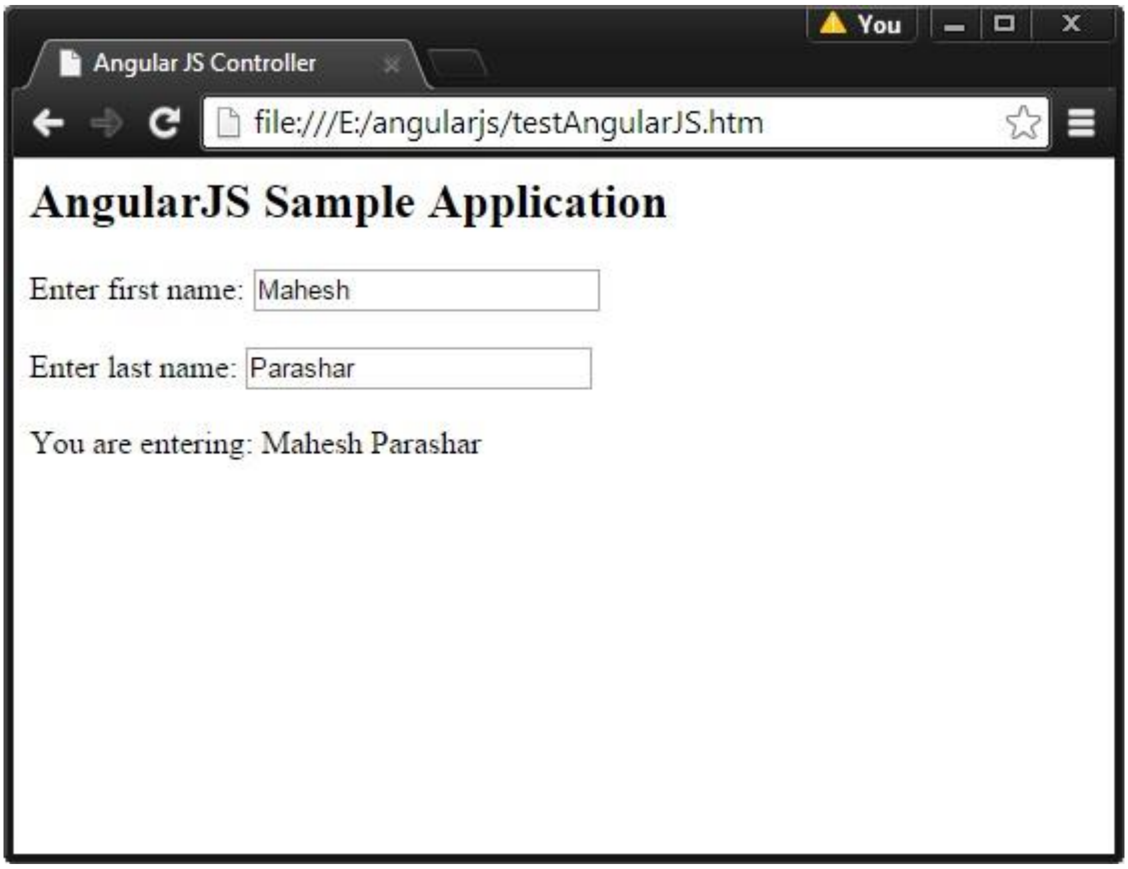
testAngularJS.htm

```
<html>

<head>

<title>Angular JS Controller</title>

</head>

<body>

<h2>AngularJS Sample Application</h2>
```

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

function studentController($scope) {

   $scope.student = {

      firstName: "Mahesh",

      lastName: "Parashar",

      fullName: function() {

         var studentObject;

         studentObject = $scope.student;

         return studentObject.firstName + " " + studentObject.lastName;

      }

   };

}

</script>

<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.15/angular.min.js">
</script>

</body>

</html>
```

Output

Open the file *testAngularJS.htm* in a web browser and see the result.



AngularJS Sample Application

Enter first name: Mahesh

Enter last name: Parashar

You are entering: Mahesh Parashar

1(g). 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>
```

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

Example

The following example shows use of all the above mentioned modules.

testAngularJS.htm

```
<html>

<head>

<title>Angular JS Modules</title>

<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
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.15/angular.min.js">
</script>

<script src="mainApp.js"></script>

<script src="studentController.js"></script>

</body>

</html>
```

mainApp.js

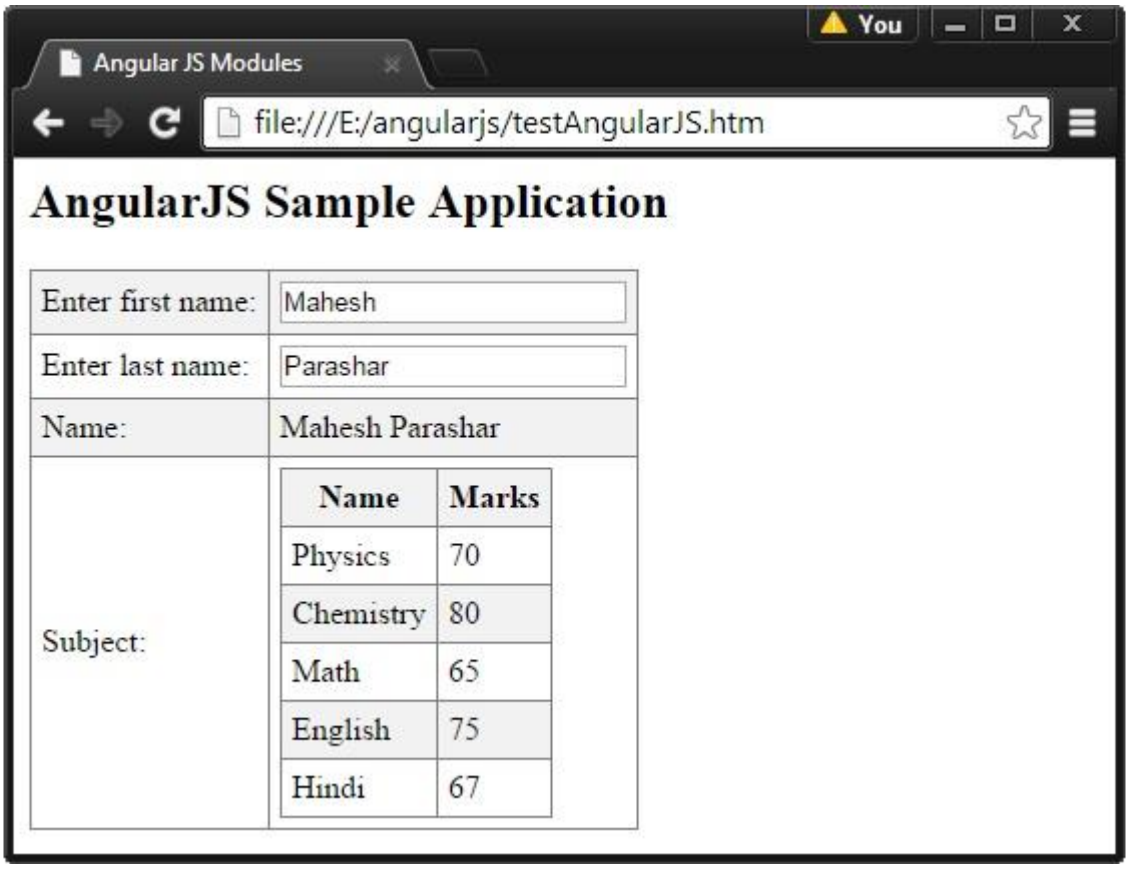
```
var mainApp = angular.module("mainApp", []);
```

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

Open the file *textAngularJS.htm* in a web browser. See the result.



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