AngularJS - Quick Guide

AngularJS - Overview

What is AngularJS?

AngularJS is an open source web application framework. It was originally developed in 2009 by Misko Hevery and Adam Abrons. It is now maintained by Google. Its latest version is 1.4.3.

Definition of AngularJS as put by its official documentation is as follows -

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. Angular's 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.

Features

- AngularJS is a powerful JavaScript based development framework to create RICH Internet Application(RIA).
- AngularJS provides developers options to write client side application (using JavaScript) in a clean MVC(Model View Controller) way.
- Application written in AngularJS is 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 and high performance web application while keeping them as easy-to-maintain.

Core Features

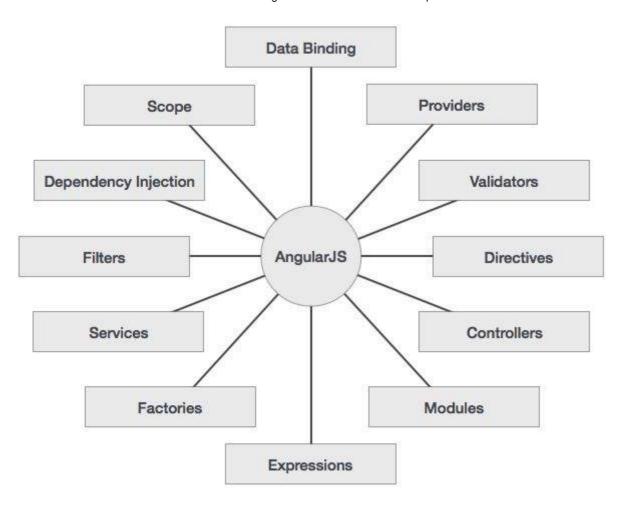
Following are most important core features of AngularJS -

- 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 that are bound to a particular scope.
- **Services** AngularJS come with several built-in services for example \$https: 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 (ngBind, ngModel...)
- **Templates** These are the rendered view with information from the controller and model. These can be a single file (like index.html) or multiple views in one page using "partials".
- Routing It is concept of switching views.
- Model View Whatever MVC 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 by making the application easier to develop, understand, and test.

Concepts

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



Advantages of AngularJS

- AngularJS provides capability to create Single Page Application in a very clean and maintainable way.
- AngularJS provides data binding capability to HTML thus giving 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, developer write less code and get more functionality.
- In AngularJS, views are pure html pages, and controllers written in JavaScript do the business processing.

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

Disadvantages of AngularJS

Though AngularJS comes with lots of plus points but same time we should consider the following points –

• **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 your application user disables JavaScript then user will just see the basic page and nothing more.

The AngularJS Components

The AngularJS framework can be divided into following 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.

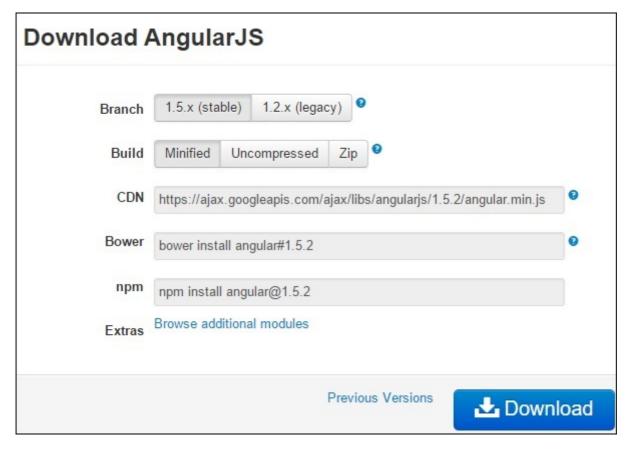
AngularJS - Environment Setup

In this chapter we will discuss about how to set up AngularJS library to be used in web application development. We will also briefly study 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 Click on this button to go to GitHub and get all of the latest scripts.
- Download AngularJS 1 Or click on this button, a screen as below would be seen –



- 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 itself are self descriptive. legacy has version less than 1.2.x and latest has 1.5.x version.
 - We can also go with the minified, uncompressed or zipped version.
 - CDN access You also have access to a CDN. The CDN will give you access
 around the world to regional data centers that in this case, Google host. This
 means using CDN moves the responsibility of hosting files from your own
 servers to a series of external ones. This also offers an advantage that if the
 visitor to your webpage has already downloaded a copy of AngularJS from the
 same CDN, it won't have to be re-downloaded.
- Try the new angularJS 2 Click on this button to download Angular JS beta 2 version. This version is very fast, mobile supported and flexible compare to legacy and latest of AngularJS 1

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* as below –

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

Following sections describe the above code in detail -

Include AngularJS

We have included the AngularJS JavaScript file in the HTML page so we can use AngularJS -

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

If you want to update into latest version of Angular JS, use the following script source or else Check the latest version of Angular JS on their official website.

```
<head>
     <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.5.2/angular.m
</head>
```

Point to AngularJS app

Next we tell what part of the HTML contains the AngularJS app. This done 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 what controller to use with this view. helloTo.title tells AngularJS to write the "model" value named helloTo.title to the 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 passed to the controller function is the *model*. 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 will see an output as below –

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

When the page is loaded in the browser, following things happen -

HTML document is loaded into the browser, and evaluated by the browser. AngularJS
 JavaScript file is loaded, the angular global object is created. Next, JavaScript which

registers controller functions is executed.

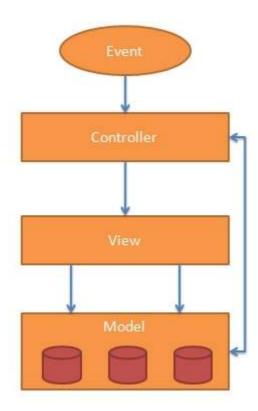
- Next AngularJS scans through the HTML to look for AngularJS apps and views. Once
 view is located, it connects that view to the corresponding controller function.
- Next, AngularJS executes the controller functions. It then renders the views with data from the model populated by the controller. The page is now ready.

AngularJS - MVC Architecture

<u>Model View Controller or MVC</u> 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.



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, we will see how AngularJS uses MVC methodology.

AngularJS - First Application

Before we start with creating actual HelloWorld application using AngularJS, let us see what are the actual 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.

Steps to create AngularJS Application

Step 1: Load framework

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

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

Step 2: Define AngularJS Application using ng-app directive

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

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

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

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

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

Steps to run AngularJS Application

Use above mentioned three steps in an HTML page.

testAngularJS.htm

Output

Open textAngularJS.htm in a web browser. Enter your name and see the result.

Sample Application	
Enter your Name:	
Hello!	

How AngularJS integrates with HTML

- ng-app directive indicates the start of AngularJS application.
- ng-model directive then creates a model variable named "name" which can be used with the html page and within the div having ng-app directive.

- ng-bind then uses the name model to be displayed in the html span tag whenever user input something in the text box.
- Closing</div> tag indicates the end of AngularJS application.

AngularJS - Directives

AngularJS directives are used to extend HTML. These are special attributes starting with ng-prefix. We're going to discuss following directives –

- **ng-app** This directive starts an AngularJS Application.
- ng-init This directive initializes application data.
- ng-model This directive binds the values of AngularJS application data to HTML input controls.
- **ng-repeat** This directive repeats html elements for each item in a collection.

ng-app directive

ng-app directive starts an AngularJS Application. It defines the root element. It automatically initializes or bootstraps the application when web page containing AngularJS Application is loaded. It is also used to load various AngularJS modules in AngularJS Application. In following example, we've defined a default AngularJS application using ng-app attribute of a div element.

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

ng-init directive

ng-init directive initializes an AngularJS Application data. It is used to put values to the variables to be used in the application. In following example, we'll initialize an array of countries. We're using JSON syntax to define array of countries.

ng-model directive

This directive binds the values of AngularJS application data to HTML input controls. In following example, we've defined a model named "name".

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

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

ng-repeat directive

ng-repeat directive repeats html elements for each item in a collection. In following example, we've iterated over array of countries.

Example

Following example will showcase all the above mentioned directives.

```
Live Demo
<html>
  <head>
     <title>AngularJS Directives</title>
  </head>
  <body>
     <h1>Sample Application</h1>
     <div ng-app = "" ng-init = "countries = [{locale:'en-US',name:'United State</pre>
        {locale: 'en-GB', name: 'United Kingdom'}, {locale: 'en-FR', name: 'France'}]"
        Enter your Name: <input type = "text" ng-model = "name">
        Hello <span ng-bind = "name"></span>!
        List of Countries with locale:
        <01>
           ing-repeat = "country in countries">
              {{ 'Country: ' + country.name + ', Locale: ' + country.locale }}
           </div>
     <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angul</pre>
     </script>
```

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

Output

Open textAngularJS.htm in a web browser. Enter your name and see the result.

Sample Application
Enter your Name:
Hello!
List of Countries with locale:
 Country: United States, Locale: en-US Country: United Kingdom, Locale: en-GB Country: France, Locale: en-FR

AngularJS - Expressions

Expressions are used to bind application data to html. Expressions are written inside double braces like {{ expression}}. Expressions behaves in same way as ng-bind directives. AngularJS application expressions are pure javascript expressions and outputs the data where they are used.

Using numbers

```
Expense on Books : {{cost * quantity}} Rs
```

Using strings

```
Hello {{student.firstname + " " + student.lastname}}!
```

Using object

```
Roll No: {{student.rollno}}
```

Using array

```
Marks(Math): {{marks[3]}}
```

Example

Following example will showcase all the above mentioned expressions.

testAngularJS.htm

```
Live Demo
<html>
   <head>
     <title>AngularJS Expressions</title>
   <body>
     <h1>Sample Application</h1>
     <div ng-app = "" ng-init = "quantity = 1;cost = 30;</pre>
         student = {firstname:'Mahesh',lastname:'Parashar',rollno:101};
        marks = [80, 90, 75, 73, 60]">
        Hello {{student.firstname + " " + student.lastname}}!
        Expense on Books : {{cost * quantity}} Rs
        Roll No: {{student.rollno}}
        Marks(Math): {{marks[3]}}
     </div>
     <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angul</pre>
     </script>
  </body>
</html>
```

Output

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

Sample Application

Hello Mahesh Parashar!

Expense on Books: 30 Rs

Roll No: 101

Marks(Math): 73

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 containing attributes/properties and functions. Each controller accepts \$scope as a parameter which refers to the application/module that controller is to control.

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

Here we've declared a controller **studentController** using ng-controller directive. As a next step we'll define the studentController as follows –

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

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

- studentController defined as a JavaScript object with \$scope as argument.
- \$scope refers to application which is to use the studentController object.
- \$scope.student is property of studentController object.
- firstName and lastName are two properties of \$scope.student object. We've passed the default values to them.
- fullName is the function of \$scope.student object whose task is to return the combined name.
- In fullName function we're getting the student object and then return the combined name.
- As a note, we can also define the controller object in 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>
<br>
You are entering: {{student.fullName()}}
```

- We've bounded student.firstName and student.lastname to two input boxes.
- We've bounded 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

Following example will showcase use of controller.

```
Live Demo -
<html>
   <head>
      <title>Angular JS Controller</title>
      <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angul</pre>
      </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"><b</pre>
         <br>
         Enter last name: <input type = "text" ng-model = "student.lastName"><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

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

AngularJS Sample Application	
Enter first name: Mahesh	
Enter last name: Parashar	
You are entering: Mahesh Parashar	

AngularJS - Filters

Filters are used to change modify the data and can be clubbed in expression or directives using pipe character. Following is the list of commonly used filters.

Sr.No.	Name & Description
1	uppercase converts a text to upper case text.
2	lowercase converts a text to lower case text.
3	currency formats text in a currency format.
4	filter filter the array to a subset of it based on provided criteria.
5	orderby 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 filter

To display only required subjects, we've used subjectName as filter.

orderby filter

To order subjects by marks, we've used orderBy marks.

```
Subject:

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

Example

Following example will showcase all the above mentioned filters.

```
Live Demo
<html>
  <head>
    <title>Angular JS Filters</title>
    <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angul</pre>
    </script>
  </head>
  <body>
    <h2>AngularJS Sample Application</h2>
    <div ng-app = "mainApp" ng-controller = "studentController">
      Enter first name:
           Enter last name: 
           <input type = "text" ng-model = "student.lastName">
```

```
Enter fees: 
                           Enter subject: 
                           <input type = "text" ng-model = "subjectName">
                  <br/>
         Name in Upper Case: {{student.fullName() | uppercase}}
                  Name in Lower Case: {{student.fullName() | lowercase}}
                  fees: {{student.fees | currency}}
                  Subject:
                           <u1>
                                             di ng-repeat = "subject in student.subjects | filter: subjects | filter: subject | filter: subject
                                                      {{ subject.name + ', marks:' + subject.marks }}
                                             </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}
                           1.
```

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

Output

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

AngularJS	Sample Application
Enter first name:	Mahesh
Enter last name:	Parashar
Enter fees:	500
Enter subject:	
Name in Lower C	Case: MAHESH PARASHAR Case: mahesh parashar
fees:	\$500.00
Subject:	Math, marks:65Physics, marks:70Chemistry, marks:80

AngularJS - Tables

Table data is normally repeatable by nature. ng-repeat directive can be used to draw table easily. Following example states the use of ng-repeat directive to draw a table.

```
Name
```

```
{td>{{ subject.marks }}
```

Table can be styled using CSS Styling.

```
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;
}
```

Example

Following example will showcase all the above mentioned directive.

```
Live Demo
<html>
   <head>
      <title>Angular JS Table</title>
      <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angul</pre>
      <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">
    Enter first name:
        <input type = "text" ng-model = "student.firstName">
      Enter last name: 
          <input type = "text" ng-model = "student.lastName">
        Name: 
        {{student.fullName()}}
      Subject:
        Name.
              Marks
            {{ subject.name }}
              {{ subject.marks }}
            </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

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

AngularJS Enter first name:	Mahesh	11
Enter last name:	Parashar	
Name:	Mahesh Parashar	
	Name	Marks
Subject:	Physics	70
	Chemistry	80
Subject.	Math	65
	English	75
	Hindi	67

AngularJS - HTML DOM

Following directives can be used to bind application data to attributes of HTML DOM Elements.

Sr.No.	Name & Description
1	ng-disabled disables a given control.
2	ng-show shows a given control.
3	ng-hide hides a given control.
4	ng-click represents a AngularJS click event.

ng-disabled directive

Add ng-disabled attribute to a HTML button and pass it a model. Bind the model to an checkbox and see the variation.

```
<input type = "checkbox" ng-model = "enableDisableButton">Disable Button
<button ng-disabled = "enableDisableButton">Click Me!</button>
```

ng-show directive

Add ng-show attribute to a HTML button and pass it a model. Bind the model to an checkbox and see the variation.

```
<input type = "checkbox" ng-model = "showHide1">Show Button
<button ng-show = "showHide1">Click Me!</button>
```

ng-hide directive

Add ng-hide attribute to a HTML button and pass it a model. Bind the model to an checkbox and see the variation.

```
<input type = "checkbox" ng-model = "showHide2">Hide Button
<button ng-hide = "showHide2">Click Me!</button>
```

ng-click directive

Add ng-click attribute to a HTML button and update a model. Bind the model to html and see the variation.

```
Total click: {{ clickCounter }}
<button ng-click = "clickCounter = clickCounter + 1">Click Me!</button>
```

Example

Following example will showcase all the above mentioned directives.

```
Live Demo
<html>
  <head>
     <title>AngularJS HTML DOM</title>
  </head>
  <body>
     <h2>AngularJS Sample Application</h2>
     <div ng-app = "">
       <input type = "checkbox" ng-model = "enableDisableButton">Disa
             <button ng-disabled = "enableDisableButton">Click Me!</button>
          <input type = "checkbox" ng-model = "showHide1">Show Button</t
             <button ng-show = "showHide1">Click Me!</button>
          <input type = "checkbox" ng-model = "showHide2">Hide Button</t
             <button ng-hide = "showHide2">Click Me!</button>
          Total click: {{ clickCounter }}
             <button ng-click = "clickCounter = clickCounter + 1">Click Me!
          </div>
     <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angul</pre>
     </script>
  </body>
</html>
```

Output

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

AngularJS Sample Application
☐ Disable Button Click Me!
☐ Show Button
☐ Hide Button Click Me!
Total click: Click Me!

AngularJS - Modules

AngularJS supports modular approach. Modules are used to separate logics say services, controllers, application etc. and keep the code clean. We define modules in separate js files and name them as per the module.js file. In this example we're going to create two modules.

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

Application Module

mainApp.js

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

Here we've declared an application **mainApp** module using angular.module function. We've passed 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,
```

Here we've declared 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've used application module using ng-app directive and controller using ng-controller directive. We've imported mainApp.js and studentController.js in the main html page.

Example

Following example will showcase all the above mentioned modules.

```
}
   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">
   Enter first name:
       <input type = "text" ng-model = "student.firstName">
     Enter last name: 
       Name: 
       {{student.fullName()}}
     Subject:
       Name
            Marks
          {{ subject.name }}
            {{ subject.marks }}
```

```
</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

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

AngularJS Sample Application

Enter first name:	Mahesh		
Enter last name:	Parashar		
Name:	Mahesh Parashar		
	Name	Marks	
	Physics	70	
Subjects	Chemistry	80	
Subject:	Math	65	
	English	75	
	Hindi	67	

AngularJS - Forms

AngularJS enriches form filling and validation. We can use ng-click to handle AngularJS click on button and use \$dirty and \$invalid flags to do the validations in seemless way. Use novalidate with a form declaration to disable any browser specific validation. Forms controls makes heavy use of Angular events. Let's have a quick look on events first.

Events

AngularJS provides multiple events which can be associated with the HTML controls. For example ng-click is normally associated with button. Following are supported events in Angular JS.

- 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();
    }
        $scope.reset();
}
```

Validate data

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

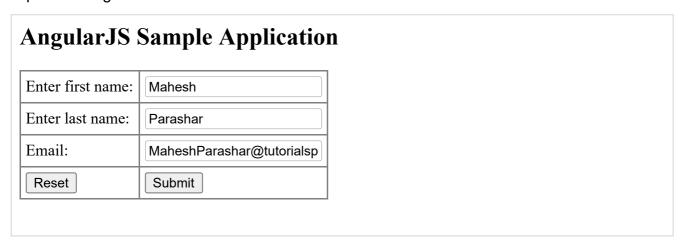
Following example will showcase all the above mentioned directives.

```
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>
        Enter first name:
             <input name = "firstname" type = "text" ng-model = "firstNa"
                <span style = "color:red" ng-show = "studentForm.firstname.$</pre>
                   <span ng-show = "studentForm.firstname.$error.required">F
                </span>
             Enter last name: 
             <input name = "lastname" type = "text" ng-model = "lastNam"
                <span style = "color:red" ng-show = "studentForm.lastname.$d</pre>
                   <span ng-show = "studentForm.lastname.$error.required">La
                </span>
             Email: <input name = "email" type = "email" ng-mod
                <span style = "color:red" ng-show = "studentForm.email.$dirt</pre>
                   <span ng-show = "studentForm.email.$error.required">Email
                   <span ng-show = "studentForm.email.$error.email">Invalid
                </span>
             <button ng-click = "reset()">Reset</putton>
             <button ng-disabled = "studentForm.firstname.$dirty &&</pre>
                   studentForm.firstname.$invalid | studentForm.lastname.$d
                   studentForm.lastname.$invalid || studentForm.email.$dirty
```

```
studentForm.email.$invalid" ng-click="submit()">Submit</br>
                 </form>
     </div>
     <script>
        var mainApp = angular.module("mainApp", []);
        mainApp.controller('studentController', function($scope) {
           $scope.reset = function() {
              $scope.firstName = "Mahesh";
              $scope.lastName = "Parashar";
              $scope.email = "MaheshParashar@tutorialspoint.com";
           }
           $scope.reset();
        });
     </script>
  </body>
</html>
```

Output

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



AngularJS - Includes

HTML does not support embedding html pages within html page. To achieve this functionality following ways are used –

- **Using Ajax** Make a server call to get the corresponding html page and set it in innerHTML of html control.
- **Using Server Side Includes** JSP, PHP and other web side server technologies can include html pages within a dynamic page.

Using AngularJS, we can embed HTML pages within a HTML page using ng-include directive.

Example

tryAngularJS.htm

```
Live Demo
<html>
   <head>
      <title>Angular JS Includes</title>
      <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angul</pre>
      </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">
         <div ng-include = "'/angularjs/src/include/main.htm'"></div>
         <div ng-include = "'/angularjs/src/include/subjects.htm'"></div>
      </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>
```

main.htm

subjects.htm

```
Subjects:

      \table \
      \table \
      \table \
      <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <th \
            <t
```

Output

To run this example, you need to deploy textAngularJS.htm, main.htm and subjects.htm to a webserver. Open textAngularJS.htm using url of your server in a web browser. See the result.

AngularJS Sample Application Enter first name: Mahesh Enter last name: Parashar

Mahesh Parashar

Subjects:

Name:

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

AngularJS - Ajax

AngularJS provides \$https: control which works as a service to read data from the server. The server makes a database call to get the desired records. AngularJS needs data in JSON format. Once the data is ready, \$https: can be used to get the data from server in the following manner

-

```
function studentController($scope,$https:) {
   var url = "data.txt";

   $https:.get(url).success( function(response) {
       $scope.students = response;
   });
}
```

Here, the file data.txt contains student records. \$https: service makes an ajax call and sets response to its property students. *students* model can be used to draw tables in HTML.

Examples

data.txt

```
{
      "Name" : "Mahesh Parashar",
      "RollNo" : 101,
      "Percentage" : "80%"
   },
   {
      "Name" : "Dinkar Kad",
      "RollNo" : 201,
      "Percentage" : "70%"
   },
   {
      "Name" : "Robert",
      "RollNo" : 191,
      "Percentage" : "75%"
   },
      "Name" : "Julian Joe",
      "RollNo" : 111,
      "Percentage" : "77%"
   }
]
```

```
<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 = "" ng-controller = "studentController">
     Name
          Roll No
          Percentage
       {{ student.Name }}
          {{ student.RollNo }}
          {{ student.Percentage }}
       </div>
  <script>
     function studentController($scope,$http) {
       var url = "data.txt";
       $http.get(url).then( function(response) {
          $scope.students = response.data;
       });
     }
  </script>
  <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.2.15/angul</pre>
  </script>
```

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

Output

To execute this example, you need to deploy *testAngularJS.htm* and *data.txt* file to a web server. Open the file *testAngularJS.htm* using the URL of your server in a web browser and see the result.

AngularJS Sample Application

Name	Roll No	Percentage
Mahesh Parashar	101	80%
Dinkar Kad	201	70%
Robert	191	75%
Julian Joe	111	77%

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

ng-view tag 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.

ng-template

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.

\$routeProvider

\$routeProvider is the key service which set the configuration of urls, map them with the corresponding html page or ng-template, and attach a controller with the same.

Usage

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'
    });

}]);
```

Following are the important points to be considered in above example.

- \$routeProvider is defined as a function under config of mainApp module using key as '\$routeProvider'.
- \$routeProvider.when defines a url "/addStudent" which then is mapped to "addStudent.htm". addStudent.htm should be present in the same path as main html page.lf htm page is not defined then ng-template to be used with id="addStudent.htm". We've used ng-template.

- "otherwise" is used to set the default view.
- "controller" is used to set the corresponding controller for the view.

Example

Following example will showcase all the above mentioned directives.

```
Live Demo
<html>
   <head>
      <title>Angular JS Views</title>
      <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angul</pre>
      <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angul</pre>
      </script>
   </head>
   <body>
      <h2>AngularJS Sample Application</h2>
      <div ng-app = "mainApp">
         <a href = "#addStudent">Add Student</a>
         <a href = "#viewStudents">View Students</a>
         <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>
```

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

AngularJS Sample Application

Add Student

View Students

Add Student

This page will be used to display add student form

AngularJS - Scopes

Scope is a special javascript object which plays the role of joining controller with the views. Scope contains the model data. In controllers, model data is accessed via \$scope object.

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

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

```
});
</script>
```

Following are the important points to be considered in above example.

- \$scope is passed as first argument to controller during its constructor definition.
- \$scope.message and \$scope.type are the models which are to be used in the HTML page.
- We've set values to models which will be reflected in the application module whose controller is shapeController.
- We can define functions as well in \$scope.

Scope Inheritance

Scope are controllers specific. If we defines nested controllers then child controller will inherit 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>
</script>
```

Following are the important points to be considered in above example.

- We've set values to models in shapeController.
- We've overridden message in child controller circleController. When "message" is used within module of controller circleController, the overridden message will be used.

Example

Following example will showcase all the above mentioned directives.

```
<body>
     <h2>AngularJS Sample Application</h2>
     <div ng-app = "mainApp" ng-controller = "shapeController">
         {{message}} <br/> {{type}} 
         <div ng-controller = "circleController">
            {{message}} <br/> {{type}} 
        </div>
        <div ng-controller = "squareController">
            {{message}} <br/> {{type}} 
        </div>
     </div>
     <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angul</pre>
     </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>
```

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

AngularJS Sample Application

In shape controller Shape

In circle controller Shape

In square controller Square

AngularJS - Services

AngularJS supports the concepts of "Separation of Concerns" using services architecture. Services are javascript functions and are responsible to do a specific tasks only. This makes them an individual entity which is maintainable and testable. Controllers, filters can call them as on requirement basis. Services are normally injected using dependency injection mechanism of AngularJS.

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

There are two ways to create a service.

- factory
- service

Using factory method

Using factory 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

Using service method, we define a service and then assign method to it. We've also injected an already available service to it.

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

Example

Following example will showcase all the above mentioned directives.

```
Live Demo
<html>
   <head>
     <title>Angular JS Services</title>
     <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angul</pre>
     </script>
   </head>
   <body>
     <h2>AngularJS Sample Application</h2>
     <div ng-app = "mainApp" ng-controller = "CalcController">
         Enter a number: <input type = "number" ng-model = "number" />
         <button ng-click = "square()">X<sup>2</sup></button>
         Result: {{result}}
     </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>
```

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

AngularJS Sample Application
Enter a number: X ²
Result:

AngularJS - Dependency Injection

Dependency Injection is a software design pattern in which components are given their dependencies instead of hard coding them within the component. This relieves a component from locating the dependency and makes dependencies configurable. This 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 simple javascript object and it is used to pass values to controller during config phase.

```
//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 value on demand whenever a service or controller requires. It normally 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 multi
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
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. Services are defined using service() functions and then injected into 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 config phase(phase during which AngularJS bootstraps itself). Below mention script can be used to create MathService that we've created earlier. Provider is a special factory method with a method get() 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
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 config phase considering the fact that value can not be used to be passed during config phase.

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

Example

Following example will showcase all the above mentioned directives.

```
Live Demo
<html>
  <head>
      <title>AngularJS Dependency Injection</title>
   </head>
   <body>
     <h2>AngularJS Sample Application</h2>
     <div ng-app = "mainApp" ng-controller = "CalcController">
         Enter a number: <input type = "number" ng-model = "number" />
         <button ng-click = "square()">X<sup>2</sup></button>
         Result: {{result}}
     </div>
     <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angul</pre>
     </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, defau
            $scope.number = defaultInput;
            $scope.result = CalcService.square($scope.number);
            $scope.square = function() {
               $scope.result = CalcService.square($scope.number);
         });
      </script>
   </body>
</html>
```

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

AngularJS Sample Application	
Enter a number: 5	
Result: 25	

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
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 sp
```

```
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

```
Live Demo
<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/angul</pre>
      </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>
```

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

AngularJS Sample Application

Student: Piyush Parashar, Roll No: 2

Student: Mahesh Parashar, Roll No: 1

AngularJS - Internalization

AngularJS supports inbuilt internationalization for three types of filters currency, date and numbers. We only need to incorporate corresponding is according to locale of the country. By default it handles the locale of the browser. For example, to use Danish locale, use following script.

```
<script src = "https://code.angularjs.org/1.2.5/i18n/angular-locale_da-dk.js">
</script>
```

Example using Danish locale

```
Live Demo
<html>
   <head>
      <title>Angular JS Forms</title>
   </head>
   <body>
      <h2>AngularJS Sample Application</h2>
      <div ng-app = "mainApp" ng-controller = "StudentController">
         {{fees | currency }} <br/>
         {{admissiondate | date }} <br/>
         {{rollno | number }}
      </div>
      <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angul</pre>
      </script>
      <script src = "https://code.angularjs.org/1.3.14/i18n/angular-locale_da-dk.</pre>
      </script>
```

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

mainApp.controller('StudentController', function($scope) {
    $scope.fees = 100;
    $scope.admissiondate = new Date();
    $scope.rollno = 123.45;
    });
    </script>

</body>
</html>
```

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

```
AngularJS Sample Application

100,00 kr

01/10/2020

123,45
```

Example using Browser's locale

```
<script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angul
</script>
<!-- <script src = "https://code.angularjs.org/1.3.14/i18n/angular-locale_a
</script> -->

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

    mainApp.controller('StudentController', function($scope) {
        $scope.fees = 100;
        $scope.admissiondate = new Date();
        $scope.rollno = 123.45;
      });
    </script>

</body>
</html>
```

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

```
AngularJS Sample Application
$100.00
Oct 1, 2020
123.45
```

AngularJS - Internationalization

AngularJS supports inbuilt internationalization for three types of filters: Currency, Date, and Numbers. We only need to incorporate corresponding java script according to locale of the country. By default, it considers the locale of the browser. For example, for Danish locale, use the following script –

```
<script src = "https://code.angularjs.org/1.2.5/i18n/angular-locale_da-dk.js">
</script>
```

Example Using Danish Locale

testAngularJS.htm

```
Live Demo
<html>
   <head>
      <title>Angular JS Forms</title>
   </head>
   <body>
      <h2>AngularJS Sample Application</h2>
      <div ng-app = "mainApp" ng-controller = "StudentController">
         {{fees | currency }} <br/>
         {{admissiondate | date }} <br/>
         {{rollno | number }}
      </div>
      <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angul</pre>
      </script>
      <script src = "https://code.angularjs.org/1.3.14/i18n/angular-locale da-dk.</pre>
      </script>
      <script>
         var mainApp = angular.module("mainApp", []);
         mainApp.controller('StudentController', function($scope) {
            $scope.fees = 100;
            $scope.admissiondate = new Date();
            $scope.rollno = 123.45;
         });
      </script>
   </body>
</html>
```

Output

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

AngularJS Sample Application 100,00 kr 01/10/2020 123,45

Example Using Browser Locale

```
Live Demo
<html>
  <head>
     <title>Angular JS Forms</title>
  </head>
   <body>
     <h2>AngularJS Sample Application</h2>
     <div ng-app = "mainApp" ng-controller = "StudentController">
         {{fees | currency }} <br/>
         {{admissiondate | date }} <br/>
         {{rollno | number }}
     </div>
     <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angul</pre>
     </script>
     <!-- <script src = "https://code.angularjs.org/1.3.14/i18n/angular-locale_a
     </script> -->
     <script>
         var mainApp = angular.module("mainApp", []);
        mainApp.controller('StudentController', function($scope) {
            $scope.fees = 100;
            $scope.admissiondate = new Date();
            $scope.rollno = 123.45;
         });
     </script>
   </body>
</html>
```

Output

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

AngularJS Sample Application

\$100.00

Oct 1, 2020

123.45