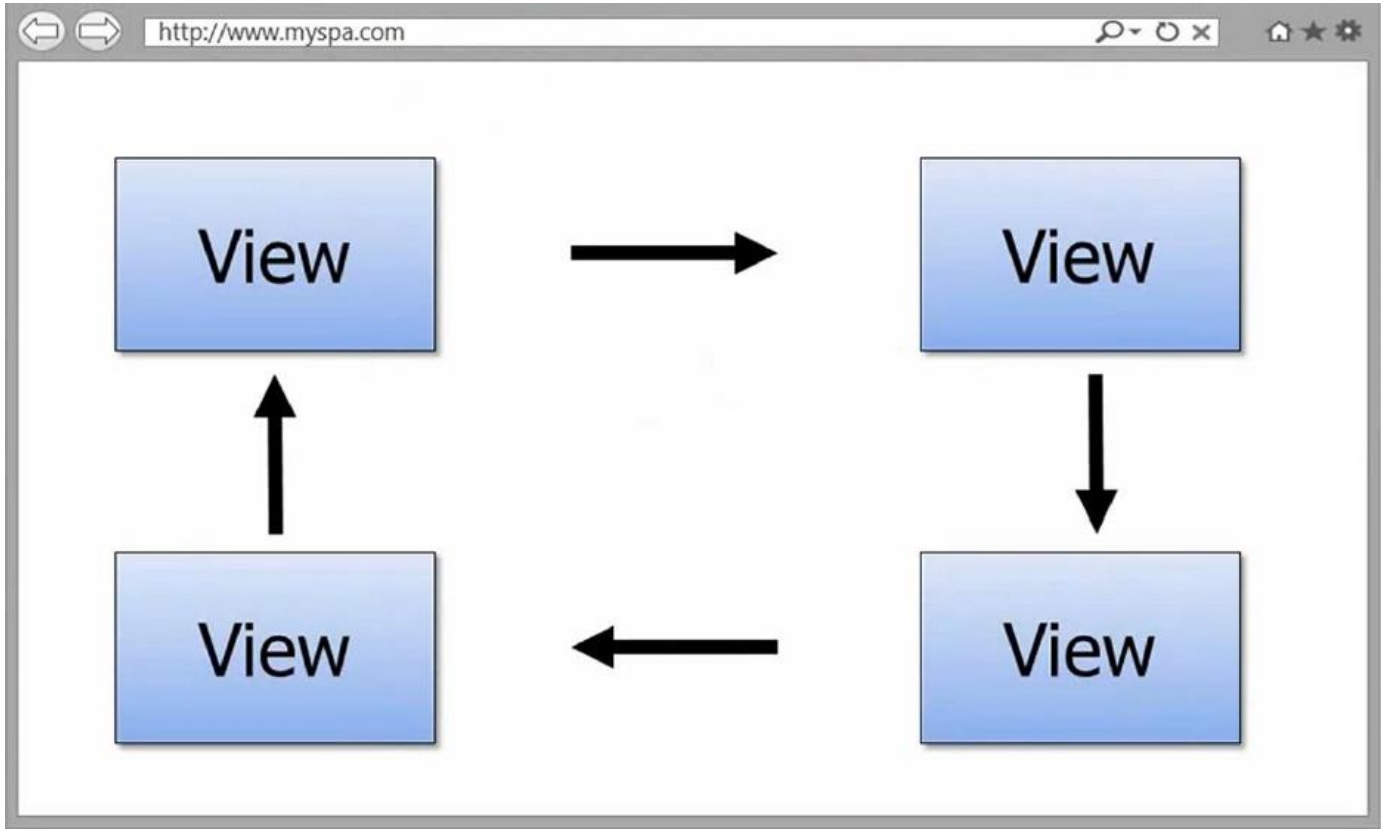
AngularJS

Single Page Application (SPA)

- Single Page Application is one in which we have a shell page and we can load multiple views into that.



SPA advantages

- In SPA, HTML page is static and all dynamic changes occur in browser. In earlier PHP, JSP, ASP, HTML was mixed with server side logic and it was generated on the server.
- SPA separates UI and data, SPA communicates with server with JSON REST API, this also allows both parts to be independently developed and tested.
- SPA is fast, as most resources HTML + CSS + Scripts are only loaded once, throughout life span of application, only data is transmitted back and forth. It Reduces bandwidth usage.
- SPA can use caching and local storage effectively.
- SPA works and feels more like an application then a web page.

SPA frameworks

- BackboneJS
- EmberJS
- AngularJS

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.

Definition of AngularJS as put by its official documentation

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.

Core Features

- 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 $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.

Core Features

- 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.

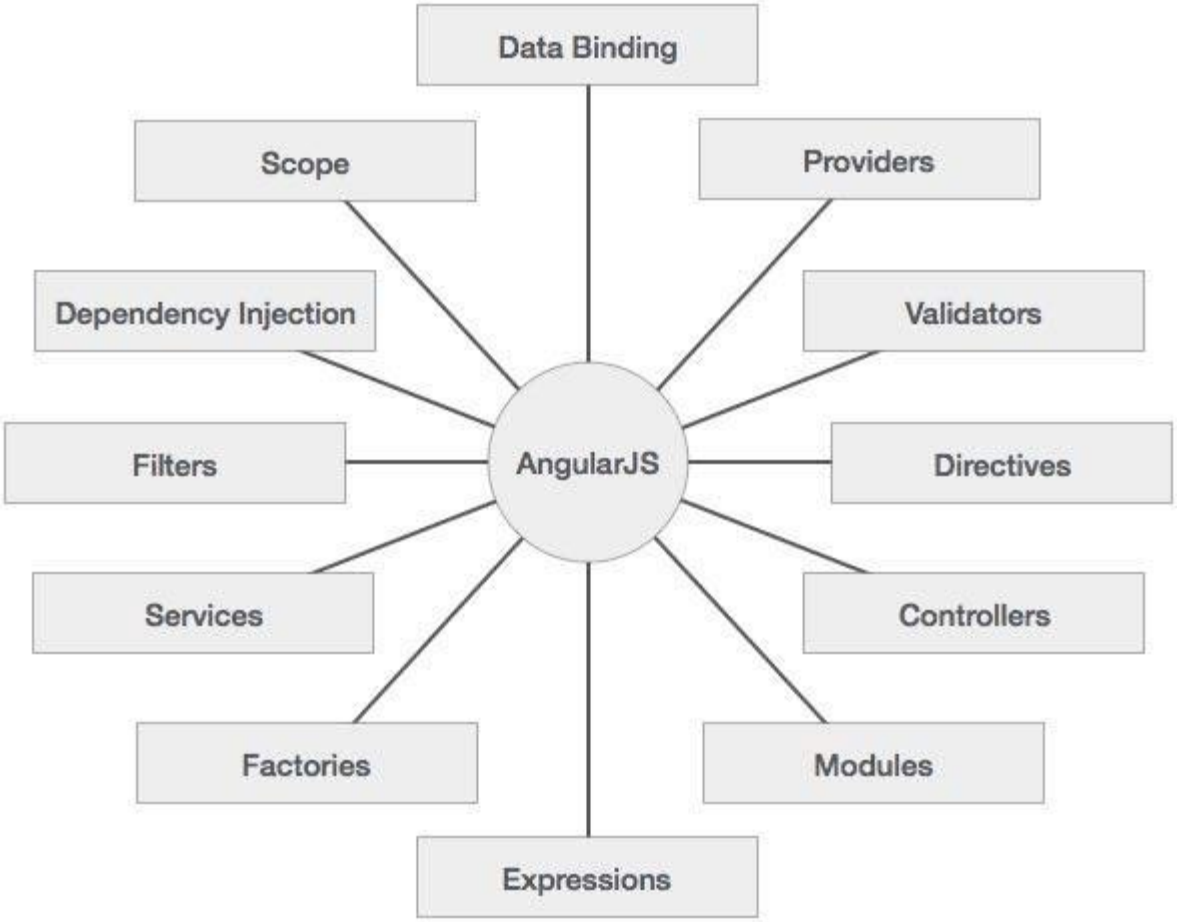
Core Features

- 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


Data Binding
Scope
Providers
Dependency Injection
Validators
Filters
AngularJS
Directives
Services
Controllers
Factories
Modules
Expressions

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.

Advantages of AngularJS

- 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 - MVC Architecture

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

- **Model** – It is the lowest level of the pattern responsible for maintaining data.

- **View** – It is responsible for displaying all or a portion of the data to the user.

- **Controller** – It is a software Code that controls the interactions between the Model and View.

Steps to create 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/1.3.14/angular.min.js "> </script>
- Step 2 – Define AngularJS Application using ng-app directive
  - ...
- 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 !

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 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 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

- 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.

```
<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

- ng-model directive defines the model/variable to be used in AngularJS Application. In following example, we've defined 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 following example, we've iterated over 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>
```

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.

AngularJS - Expressions

- 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]}}

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>
```

studentController

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

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

studentController

- 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 defined the controller object in separate JS file and refer that file in the html page.

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.

Filters

- 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 Upper Case: {{student.fullName() | lowercase}}
```

Filters

- 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.

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

Filters

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

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

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.

HTML DOM

- 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>
```

HTML DOM

- 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.

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

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.

Application Module

- **Application Module** used to initialize an application with controller(s).

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

Controller Module

```
mainApp.controller("studentController", function($scope) {
        $scope.student = {
                firstName: “Onkar",
                lastName: “Deshpande",
                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;
                }
        };
});
```

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.

Events

- 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

```
<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 = “Onkar";
                                    $scope.lastName = “Deshpande";
                                    $scope.email = “onkar.java@gmail.com";
                        }
                        $scope.reset();
            }
</script>
```

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.

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.

```
<div ng-app = "" ng-controller = "studentController">
            <div ng-include = "'main.htm'"></div>
            <div ng-include = "'subjects.htm'"></div>
</div>
```

AngularJS - Ajax

- AngularJS provides $http 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, $http can be used to get the data from server in the following manner –

```
function studentController($scope,$http) {
        var url = "data.txt";
        $http.get(url).success( function(response) {
                $scope.students = response;
        });
}
```

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.

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

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.

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

$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.

```
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' });
}]);
```

Important points to be considered

- $routeProvider is defined as a function under config of mainApp module using key as '$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.If 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.

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.

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

Important points to be considered

- $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>
```

Important points to be considered

- 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.

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.

Inbuilt Services

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

2 Ways to create Services

- factory method

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

- service method

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

Custom directives

AngularJS provides series of inbuilt directives. However, sometimes we need to write our own directive called as ‘custom directive’.