Syllabus ...

1. AngularJS Core Concepts [8 lectures]

1.1 What is AngularJS?

1.2 Difference between Javasript and Angular JS

1.3 Advantages of Angular

1.4 AngularJS MVC Architecture

1.5 Introduction to SPA

1.6 Setting up the environment

1.7 First App using MVC architecture

2. AngularJS Directives and Expressions [10 lectures]

2.1 Understanding ng attributes: ng-app, ng-init, ng-model, ng-controller, ng-bind, ng-repeat, ng-show, ng-readonly, ng-disabled, ng-if, ng-click

2.2 Expression and Data Binding

2.3 Working with directives

3. AngularJS Modules, Controller, View and Scope [10 lectures]

3.1 Angular Modules

3.2 Angular Controller

3.3 Angular View

3.4 Scope Hierarchy

4. Filter, Forms and Ajax Filters [12 lectures]

4.1 Built-in filter - upper case and lower case filters, date, currency and number formatting orderBy, filter, custom filter,

4.2 Angular JS Forms – Working with AngularJS forms, model binding, form controller, Using CSS classes, form events, custom model update triggers, custom validation, $http service,

4.3 Ajax implementation using $http

5. Dependency Injection, Services [8 lectures]

5.1 What is dependency injection?

5.2 Understanding services

5.3 Using built-in service

5.4 Creating custom service,

5.5 Injecting dependency in service

Contents ...

***

1...

AngularJS Core Concepts

Objectives...

- To understand the concept of AngularJS.
- To learn difference between JavaScript and AngularJS.
- To study AngularJS MVC Architecture.
- To understand the concept of SPA.
- To develop first App using MVC architecture.

1.1 INTRODUCTION

- AngularJS is a JavaScript framework, which is used to build a Web Application. Google has developed AngularJS. AngularJS is an open source project i.e. it can be downloaded, used and changed by anyone as per the requirements.
- AngularJS provides an excellent platform for creating Single Page Application (SPA) and rich Web Applications. Example of a Single Page Application is Gmail.
- Angular interprets attributes as directives to bind input or output parts of the page to a model that is represented by standard JavaScript variables. The values of those JavaScript variables can be manually set within the code, or retrieved from static or dynamic JSON resources.

1.2 WHAT IS AngularJS ?

- AngularJS is a JavaScript framework to create user rich single page web applications.
- AngularJS was developed by Misko and Adam Abrons in 2009 in order to help combat such issues and is being maintained by Google. It is an open source project which means you can freely use and share it.
- AngularJS is a JavaScript framework which is based on Google's JS engine. It is used for creating pretty UI and fast apps.

- AngularJS is a client-side JavaScript MVC framework to develop a dynamic we application. It was originally started as a project in Google but now, it is open sourc framework.
- AngularJS is entirely based on HTML and JavaScript, so there is no need to lear another syntax or language.
- AngularJS changes static HTML to dynamic HTML. It extends the ability of HTML b adding built-in attributes and components and also provides an ability to creat custom attributes using simple JavaScript.
- AngularJS is popular among developers. For single-page applications. AngularJ framework creates rich interactive features for a real-time experience.

1.2.1 AngularJS Lifecycle

- AngularJS Lifecycle has three phases: Bootstrap, Compilation, and Runtime.
- To understand how to design and implement our code we need to understanding th lifecycle of an AngularJS application.

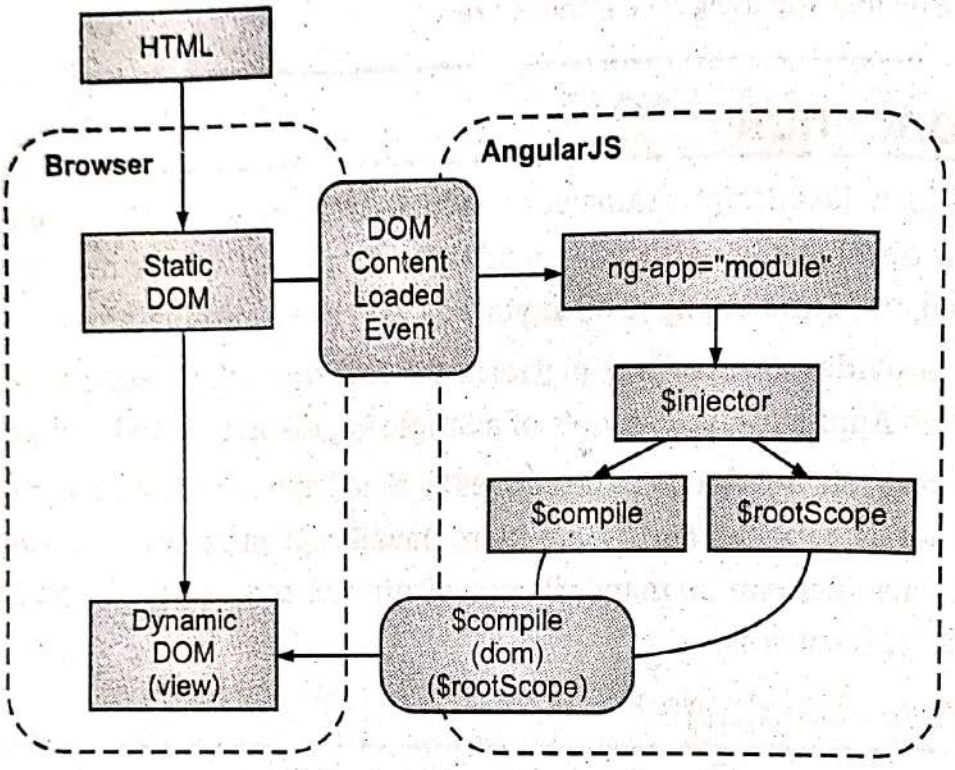


Fig. 1.1: AngularJS Lifecycle

- The following sections describe these phases of an AngularJS application.

1. **The Bootstrap Phase**
   - The first phase of the AngularJS lifecycle is the bootstrap phase, which occurs when the AngularJS JavaScript library is downloaded to the browser.
   - AngularJS initializes its own necessary components and then initializes our module, which the ng-app directive points to.
   - The module is loaded, and any dependencies are injected into our module and made available to code within the module.

2. **The Compilation Phase**
   - The second phase of the AngularJS lifecycle is the HTML compilation stage. Initially when a web page is loaded, a static form of the DOM is loaded in the browser.
   - During the compilation phase, the static DOM is replaced with a dynamic DOM that represents the AngularJS view.
   - This phase involves two parts: Traversing the static DOM and collecting all the directives and then linking the directives to the appropriate JavaScript functionality in the AngularJS built-in library or custom directive code. The directives are combined with a scope to produce the dynamic or live view.
3. **The Runtime Data Binding Phase**
   - The final phase of the AngularJS application is the runtime phase, which exists until the user reloads or navigates away from a web page. At that point, any changes in the scope are reflected in the view, and any changes in the view are directly updated in the scope, making the scope the single source of data for the view.
   - AngularJS behaves differently from traditional methods of binding data. AngularJS compiles the DOM only once and then links the compiled template as necessary, making it much more efficient than traditional methods.

1.2.2 AngularJS 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 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 return 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 ng-bind, ng-model, 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 Controller:** 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-View Model). The Angular JS team refers it humorously as Model View Controller.

- **Deep Linking:** Deep linking allows 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.

1.3 DIFFERENCE BETWEEN JavaScript AND AngularJS

- Following Table 1.1 shows the difference between JavaScript and AngularJS.

Table 1.1: Difference between JavaScript and AngularJS

JavaScript	AngularJS
1. JavaScript is an object-oriented script language.	1. AngularJS is an open source framework for creating dynamic web applications.
2. It is used for creating dynamic web applications.	2. It is used for creating large single page applications.
3. It was mainly developed by Netscape communications.	3. It was mainly developed by Google.
4. JavaScript based on concept of dynamic typing as an interpreted language.	4. It is based on concept of model view controller to build apps.
5. Its syntax is complex as compared to AngularJS.	5. Its syntax is simple.
6. It is complex to learn.	6. It is easy to learn.
7. It does not support filters.	7. It supports filters.
8. It validates the user input at the browser level before submitting the page to the sever end.	8. It makes an ideal tool for any server technology.
9. It's a full featured language used to manipulate Document Object Model (DOM).	9. It extends HTML with new attributed along with other web technologies.

1.4 ADVANTAGES AND DISADVANTAGES OF AngularJS

Advantages of AngularJS:

- **Built by Google:** AngularJS has been developed as well as maintained by dedicated Google engineers. The various advantages of Google supported sites are regular updates are done. For distributed or remote users, the anywhere/anytime access capability to the corporate intranet by Google supported sites. It provides a capacity to work across the operating system.
- **Great MVC:** As we know there are three components of MVC architecture (Model, View, Controller) so in many frameworks, a programmer has to split the code into multiple MVC components. Also, after that programmer has to code again to combine the code of these three parts. While in angular it is being done automatically. Angular strings the code together and hence saves the time of programmer too.

- **Intuitive:** AngularJS is more intuitive as it ma language. Moreover, it is less brittle for reorganizi
- **Open Source:** AngularJS is an open source JavaSc advantage of easily changing the source code to any changes to satisfy the customer requireme employment.
- **Comprehensive:** AngularJS is a comprehen development. It does not need any other plugin range of other features that include Restfu injection, enterprise-level testing, etc.
- **Reusable:** AngularJS provides reusable compo
- **Single Page Application (SPA):** Single page web page is loaded and further updation is mostly used to create single page applicatio as well as it is user-friendly.

Disadvantages of AngularJS:

- **Confusion:** There are multiple ways to do th it can be hard for novices to say which way for programmers to develop an understan they help.
- **Lagging UI:** If there are more than 2000 This means that the possible complexity o data grids and lists.
- **Not Secure:** Being JavaScript only fram not safe. Server-side authentication application secure.
- **Not degradable:** If the user of your would be visible, except the basic page.

1.5 AngularJS MVC ARCHITECT

- **Model View Controller** or MVC is a applications. AngularJS follows the framework as shown below.

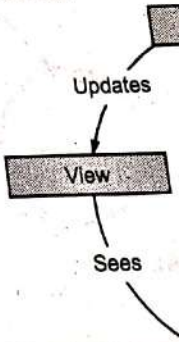


Fig. 1.2: An

- **Intuitive:** AngularJS is more intuitive as it makes use of HTML as a declarative language. Moreover, it is less brittle for reorganizing.
- **Open Source:** AngularJS is an open source JavaScript MVC framework. It provides the advantage of easily changing the source code to provide clarification. We can make any changes to satisfy the customer requirements. It can add employment or delete employment.
- **Comprehensive:** AngularJS is a comprehensive solution for rapid front-end development. It does not need any other plugins or frameworks. Moreover, there are a range of other features that include Restful actions, data building, dependency injection, enterprise-level testing, etc.
- **Reusable:** AngularJS provides reusable components.
- **Single Page Application (SPA):** Single page application means only a single HTML web page is loaded and further updation is done on that single page only. Since it is mostly used to create single page application and single page application works fast as well as it is user-friendly.

Disadvantages of AngularJS:

- **Confusion:** There are multiple ways to do the same thing with AngularJS. Sometimes, it can be hard for novices to say which way is better for a task. Hence, it is imperative for programmers to develop an understanding of the various components and how they help.
- **Lagging UI:** If there are more than 2000 watchers, it can get the UI to severely lag. This means that the possible complexity of Angular Forms is limited. This includes big data grids and lists.
- **Not Secure:** Being JavaScript only framework, application written in AngularJS are not safe. Server-side authentication and authorization are 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.

1.5 AngularJS MVC ARCHITECTURE

- **M**odel **V**iew **C**ontroller or MVC is a software design pattern for developing web applications. AngularJS follows the MVC architecture, the diagram of the MVC framework as shown below.

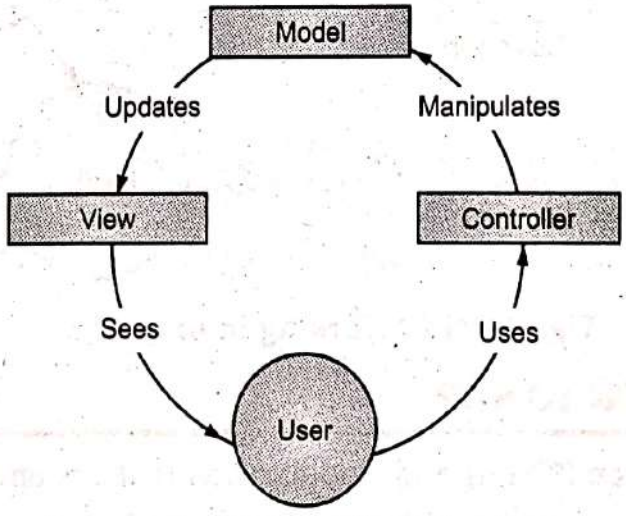


Fig. 1.2: AngularJS Architecture Diagram

- **Model:** Models are used to represent your data. A model in AngularJS is a primitive data type such as Number, String, Boolean, Object, etc. It is the lowest level of the pattern responsible for maintaining data. It is a simple JavaScript object without any getter and setter methods.
- **View:** In AngularJS, Document Object Model (DOM) is what users see. It is responsible for displaying all or a portion of the data to the user.

  Views are used to represent the presentation layer which is provided to the end users. In order to display the data from controller, Angular expressions can be added to the view which will coordinate model and view about any modification.
- **Controller:** Controller is a collection of JavaScript classes where application logic is defined. It is a software code that controls the interactions between the Model and View. Model resides inside this controller. The Controller represents the layer that has the business logic. User events trigger the functions which are stored inside your controller. The user events are part of the controller.

How MVC Works in AngularJS?

- We can implement MVC pattern through JavaScript and HTML. HTML helps to implement model part, while JavaScript is for view and controller part.
- When a user enters a URL in the browser, according to that specific URL control goes to the server and calls the controller. After that controller uses the appropriate model and appropriate view so that appropriate response can be made for user's request. It creates the response and that response is sent back to the user.

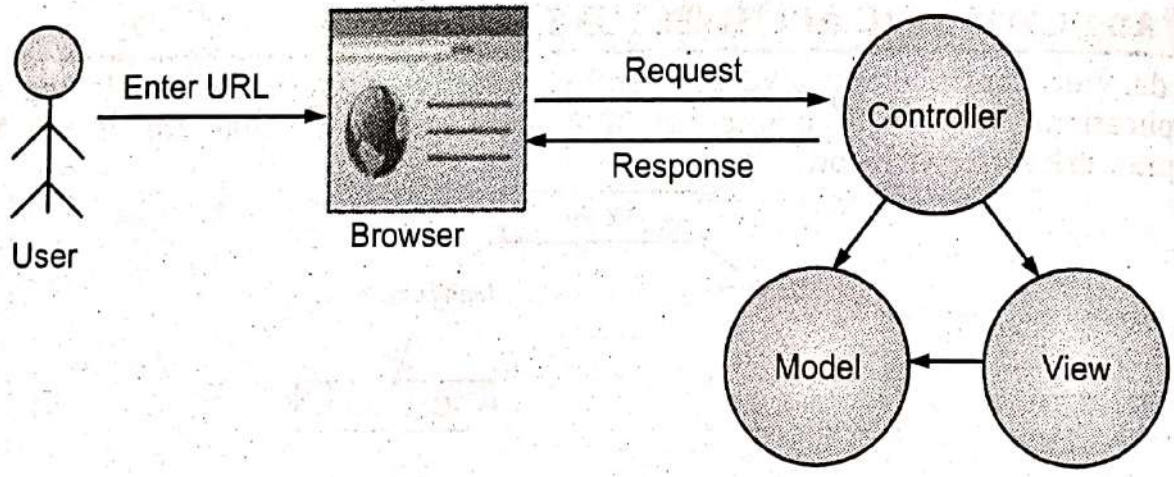


Fig. 1.3: MVC Working in AnguarJS

1.6 INTRODUCTION TO SAP

- Single Page Application (SPA) is a web application that fits on a single page. All your code (JS, HTML, CSS) is retrieved with a single page load.

- AngularJS is one of the powerful frame work tool.
- SPA uses the concept of routing in which main page remains the same and navigation between pages/views are performed without reloading the main page. The content from other view pages are fetched and loaded in the HTML tag where ng-view directive has been defined (we will be learn in next chapter).
- Single Page Applications are Web Applications that load the HTML and dynamically keep on refreshing the required portion of the page with the newly updated data.

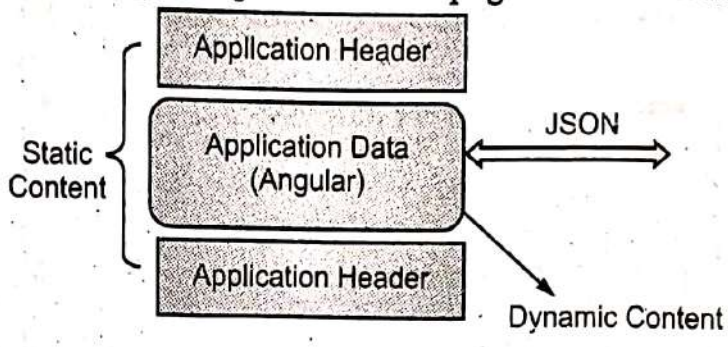


Fig. 1.4: Single Page Application

- Single page application avoid loading of the full page and load only the portion of the page with the required and the meaningful data, thus the SPAs are very fast.

Advantages of SPA:

- Single Page Application is good for making Responsive Websites.
- The whole page need not be refreshed.
- In the SPA, all the required resources like HTML, CSS, and Scripts loaded only once which makes application fast.
- Provides the better experience to user.
- It allows us to work offline.

Disadvantages of SPA:

- The initial loading is quite slow.
- It requires more resources to be loaded.
- The client should have JS enabled.
- It is complex to build and develop.

1.7 SETUP AngularJS DEVELOPMENT ENVIRONMENT

- We need the following tools to setup a development environment for AngularJS:
  1. AngularJS Library
  2. Editor/IDE
  3. Browser
  4. Web Server

1. **AngularJS Library:** We need to download AngularJS library file from *angularjs.org*.

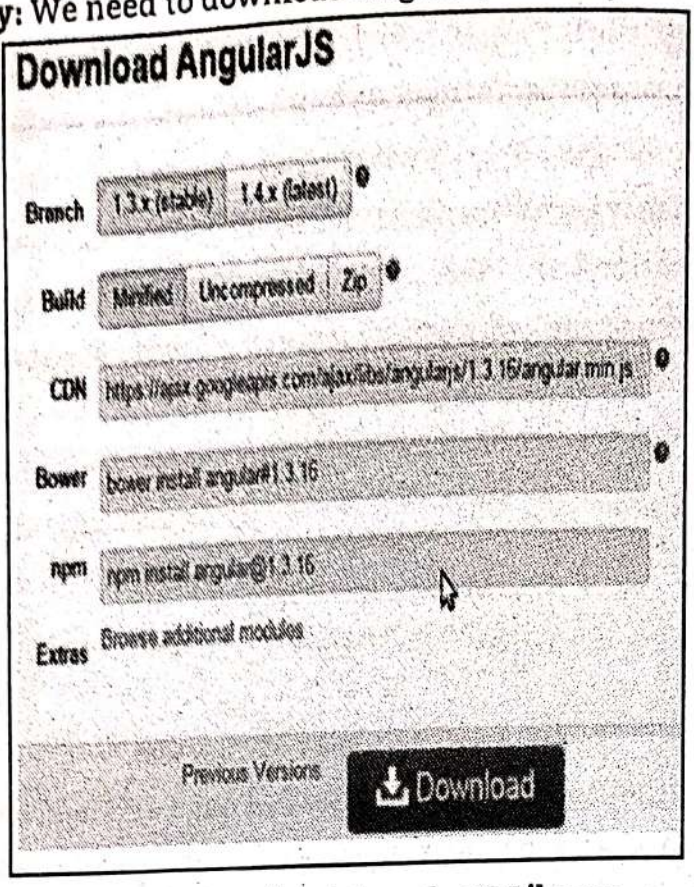


Fig. 1.5: Download AngularJS Library

2. **Editor:** It is an HTML and JavaScript code. So, we can install any editor/IDE as per customer choice. Here are the popular editors like Visual Studio, Sublime Text, Notepad, etc.
3. **Browser:** AngularJS is compatible with any web browsers like Google, Firefox, etc.
4. **Web Server:** AngularJS apps need a web server to process data. Therefore, we use web servers like Apache, IIS, etc. to set up AngularJS environment.

1.8 FIRST AngularJS APPLICATION

- We will create a simple AngularJS web application step by step and understand the basic building blocks of AngularJS.

1. Create an HTML document with <head> and <body> elements, as shown below.

```
<!DOCTYPE html>

<html>
  <head>

  </head>
  <body>

  </body>
</html>
```

2. Include angular.js file in the head section.

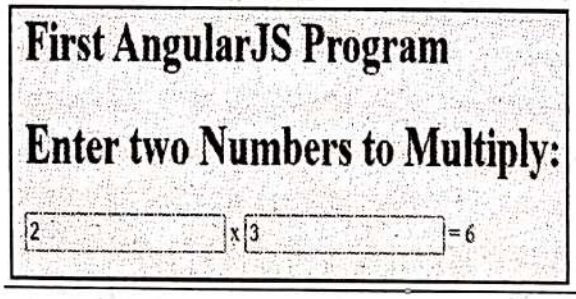
```
<!DOCTYPE html>
<html>
<head>
    <title>First AngularJS Application</title>
    <script src= ".....Scripts/angular.js"></script>
</head>
<body>

</body>
</html>
```

- Here, we will be creating a simple multiplier application which will multiply two numbers and display the result. User will enter two numbers in two separate textboxes and the result will be displayed immediately.

Program 1.1: AngularJS program for multiplication of two numbers.

```
<!DOCTYPE html>
<html>
<head>
<title> First AngularJS Program </title>
   <script
   src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.
   js">
   </script>
 </head>
 <body ng-app >
      <h1>First AngularJS Program</h1>
      <h1> Enter two Numbers to Multiply: </h1>
      <input type="text" ng-model="Num1" /> x <input type="text" ng-
      model="Num2" /> = <span>{{Num1 * Num2}}</span>
  </body>
</html>
```

Output:



In the above program,

1. The **ng-app** directive defines an AngularJS application.
2. The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data.
3. **Expression** is like JavaScript code which is usually wrapped inside double curly braces such as {{ expression }}. AngularJS framework evaluates the expression and produces a result. In the above example, {{ Num1 * Num2}} will simply display the product of Num1 and Num2.

1.9 FIRST APP USING MVC ARCHITECTURE

MVC in AngularJS

- Here, we will see how the components of MVC Architecture work in AngularJS. Refer following program to see Model, View and Controller parts in AngularJS program.

(i) Model part in Angular Application

It responds to the request made from view part. It can obtain data dynamically means you can get data from database like from MySQL or you can get data from static JSON (JavaScript Object Notation) file. For example,

```
$scope.helloToAll = {};
$scope.helloToAll = "SYBBA-CA students Welcome you all!!!!!";
```

(ii) View part in Angular Application

It is the presentation part of an application. Through the view part, a user can see the Model part. Therefore, the model and view part join together.

For example,

```
<h2>Hello {{helloToAll}} !</h2>
```

(iii) Controller part in Angular Application

It is the central part of the angular application. Everything starts with a controller in angular. The model part and the view part join with each other. It means whatever happens to the model part automatically effects to the view part and whatever happens to the view part automatically effects to the model part also. Controller joins the model and view part.

For example,

```
.controller("MyController", function($scope)
{
    $scope.helloToAll = {};
    $scope.helloToAll = "SYBBA-CA students Welcome you all!!!!!";
} );
```

Program 1.2: Program with a model part, view part and controller part in the single code.

```
<!DOCTYPE html>
<html>
<head>
 <script
 src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js
 ">
 </script>
 </head>
<body ng-app="myapp">
<div ng-controller="MyController" >
<h2>Hello {{helloToAll}} !</h2>
</div>
<script>
  angular.module("myapp", [])
  .controller("MyController", function($scope)
  {
     $scope.helloToAll = {};
     $scope.helloToAll = "SYBBA-CA students Welcome you all!!!!!";
  } );
</script>
</body>
</html>
```

Output:

Hello SYBBA-CA students Welcome you all!!!!! !

Summary

AngularJS is a JavaScript framework to create user rich single page web applications.
AngularJS is a client-side JavaScript MVC framework to develop a dynamic web application.
AngularJS was developed by Misko and Adam Abrons in 2009 in order to help combat such issues and is being maintained by Google. It is an open source project which means you can freely use and share it.

- AngularJS changes static HTML to dynamic HTML. It extends the ability of HTML by adding built-in attributes and components and also provides an ability to create custom attributes using simple JavaScript.
- AngularJS has various features like Data Binding, Scope, Controller, Services, Filters, Directives, Templates, Routing, Model View Controller, Deep Liking, Dependency Injection, etc.
- Model View Controller or MVC as it is popularly called, is a software design pattern for developing web applications.
- In MVC architecture the model is the data, the view is the User Interface and the controller is the business logic.
- Single page application (SPA) is a web application that fits on a single page.
- We need the following tools to setup a development environment for AngularJS: AngularJS Library, Editor/IDE, Browser, Web server.

Check Your Understanding

1. AngularJS is based on the ..... pattern.
   (a) VMC (b) MVC
   (c) MCV (d) CVM
2. What is Angular Js?
   (a) Library (b) Framework
   (c) Plugin (d) Browser Extension
3. AngularJS applications are a mix of .....
   (a) HTML and PHP (b) HTML and CrossScript
   (c) HTML and AngularScript (d) HTML and JavaScript
4. Who is sometimes called as Father of AngularJS?
   (a) Brad Green (b) Igor Minor
   (c) MiskoHevery (d) Brian Ford
5. How to include HTML content into another HTML?
   (a) Server Side push (b) Use of CommetD
   (c) Use of Polling (d) Use of include
6. What is MVC (Mode View Controller)?
   (a) It is a service for Java built function.
   (b) It is marker for DOM elements.
   (c) It is a software design pattern for developing web applications.
   (d) None of the above

7. Which of the following is true about Ang
   (a) Services are JavaScript functions.
   (b) Services are responsible to do specif
   (c) Inbuilt services are always prefixed
   (d) All of the above
8. Which of the following is valid for Ang
   (a) `var app = angular.module(["myA`
   (b) `var app = angular.module("myA`
   (c) `var app = angular.module();`
   (d) `var app = angular.module("my`
9. What angular function is used to m
   (a) angular.bootstrap
   (c) angular.copy
10. Which of the following sentence i
    (a) `$routeProvider` maps `Urls` wi
    (b) `$routeProvider` attaches a co
    (c) `$routeProvider` is the key se
    (d) All of the above

1. (b)	2. (b)	3. (d)	4. (c)	5

Practice Questions

Q. I Answer the following questio

1. What is AngularJS?
2. What is SPA?
3. What are the tools to setup
4. What are the main featur
5. How MVC works in Angul
6. Explain use of Model par
7. Explain use of View part
8. Explain use of Controlle

7. Which of the following is true about AngularJS service?
   (a) Services are JavaScript functions.
   (b) Services are responsible to do specific tasks only.
   (c) Inbuilt services are always prefixed with $ symbol.
   (d) All of the above
8. Which of the following is valid for AngularJS module?
   (a) `var app = angular.module(["myApp","param");`
   (b) `var app = angular.module("myApp", []);`
   (c) `var app = angular.module();`
   (d) `var app = angular.module("myApp");`
9. What angular function is used to manually start up an angular application?
   (a) angular.bootstrap (b) angular.element
   (c) angular.copy (d) None of the above
10. Which of the following sentence is true about `$routeProvider`?
    (a) `$routeProvider` maps `Urls` with the corresponding html page or ng-template.
    (b) `$routeProvider` attaches a controller with the view.
    (c) `$routeProvider` is the key service which set the configuration of urls.
    (d) All of the above

ANSWERS

1. (b)	2. (b)	3. (d)	4. (c)	5. (d)	6. (c)	7. (d)	8. (d)	9. (a)	10. (d)

Practice Questions

I Answer the following questions in short.

1. What is AngularJS?
2. What is SPA?
3. What are the tools to setup a development environment for AngularJS?
4. What are the main features of AngularJS?
5. How MVC works in AngularJS?
6. Explain use of Model part in MVC Architecture?
7. Explain use of View part in MVC Architecture?
8. Explain use of Controller part in MVC Architecture?

Q. II Answer the following questions.

1. Explain what are the key features of AngularJS?
2. What are the advantages of using AngularJS?
3. Explain difference between AngularJS and JavaScript.
4. Explain the AngularJS MVC Architecture.
5. What are the disadvantages of AngularJS?
6. Write a short note on SPA.
7. Write a program to display name and address using MVC architecture.

Q. III Define the following terms.

1. Model View Controller
2. Data Binding.

2... AngularJS Directives and Expressions

Objectives...

- To understand different ng attributes.
- To learn Expression and Data Binding.
- To understand the working with directives.

2.1 INTRODUCTION

A directive is a rule for defining how your UI interacts with data binding.

Directives are special attributes starting with ng- prefix.

Data binding is a very useful and powerful feature used in software development technologies. It acts as a bridge between the view and business logic of the application.

Expressions in AngularJS are used to bind application data to HTML.

2.2 UNDERSTANDING ng ATTRIBUTES

Directives are markers on a DOM element that tell AngularJS to attach a specified behaviour to that DOM element or even transform the DOM element and its children. In short, it extends the HTML.

We use attribute directives to apply conditional style to elements, show or hide elements or dynamically change the behaviour of a component according to a changing property.

Most of the directives in AngularJS are starting with ng. AngularJS includes various built-in directives. In addition to this, you can create custom directives for your application.

AngularJS facilitates you to extend HTML with new attributes. These attributes are called directives.

- ng prefix on your own directives in order to avoid possible name collisions in futu versions of Angular.

Following are the most common built-in AngularJS directives.

1. **ng-app:**

- The ng-app directive in AngularJS is used to define the root element of an Angular application. This directive automatically initializes the AngularJS application on pag load. It can be used to load various modules in AngularJS Application.

Syntax:

```
<element ng-app=""> Contents... </element>
```

Program 2.1: Program to demonstrate ng-app.

```
<!DOCTYPE html>
<html>
<head>
<script src= "https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angula
</script>
</head>
<body style="text-align:center">
<h2>To demostrate ng-app directive</h2>
<div ng-app="" ng-init="name='Hi all'">
<p>{{ name }} Welcome you all.</p>
</div>
</body>
```

Output:

To demostrate ng-app directive

Hi all Welcome you all.

2. **ng-init:**

- The ng-init directive can be used to initialize variables in AngularJS application.

Program 2.2: The following program demonstrates ng-init directive that initialize variable of string, number, array, and object.

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.5/angular.min.js">
</script>
</head>
```



```
<body>
<div ng-app ng-init="greet='
amount= 10; myArr = [100, 30
person = { firstName:'Archa
Amount= {{amount}} <br />
Array index 0 value={{myArr
Array index 1 value={{myAr
First Name={{person.firstN
Last Name= {{person.lastNa
</div>
</body>
</html>
```

Output:

Amount= 10
Array index 0 value=100
Array index 1 value=300
First Name=Archana
Last Name= Kothawade

3. **ng-repeat:**

- Angular-JS ng-repeat di times or once per item i objects.
- If an item is remove automatically, adjustin
- The "ng-repeat" direct iterating the collectio
- These variables are sh

Syntax:

```
<div ng-repeat="Nam
  {{Name}}
</div>
```

Here "ObjectName" each value inside it usi

```
<body>
<div ng-app ng-init="greet='Hello World!';
amount= 10; myArr = [100, 300];
person = { firstName:'Archana', lastName :'Kothawade'}">
Amount= {{amount}} <br />
Array index 0 value={{myArr[0]}} <br />
Array index 1 value={{myArr[1]}} <br />
First Name={{person.firstName}}<br />
Last Name= {{person.lastName}}
</div>
</body>
</html>
```

Output:

Amount= 10
Array index 0 value=100
Array index 1 value=300
First Name=Archana
Last Name= Kothawade

3. ng-repeat:

- Angular-JS ng-repeat directive is used to repeat a set of HTML code for a number of times or once per item in a collection of items. ng-repeat is mostly used on arrays and objects.
- If an item is removed from the source Array, Angular will update the DOM automatically, adjusting all index values accordingly.
- The "ng-repeat" directive has a set of unique variables that are useful while iterating the collection.
- These variables are shows in the following table.

Syntax:

```
<div ng-repeat="Name in ObjectName ">
  {{Name}}
</div>
```

Here "ObjectName" is a collection that can be an object or an array and you can access each value inside it using a "Name".

Table 2.1: ng-repeat Variables

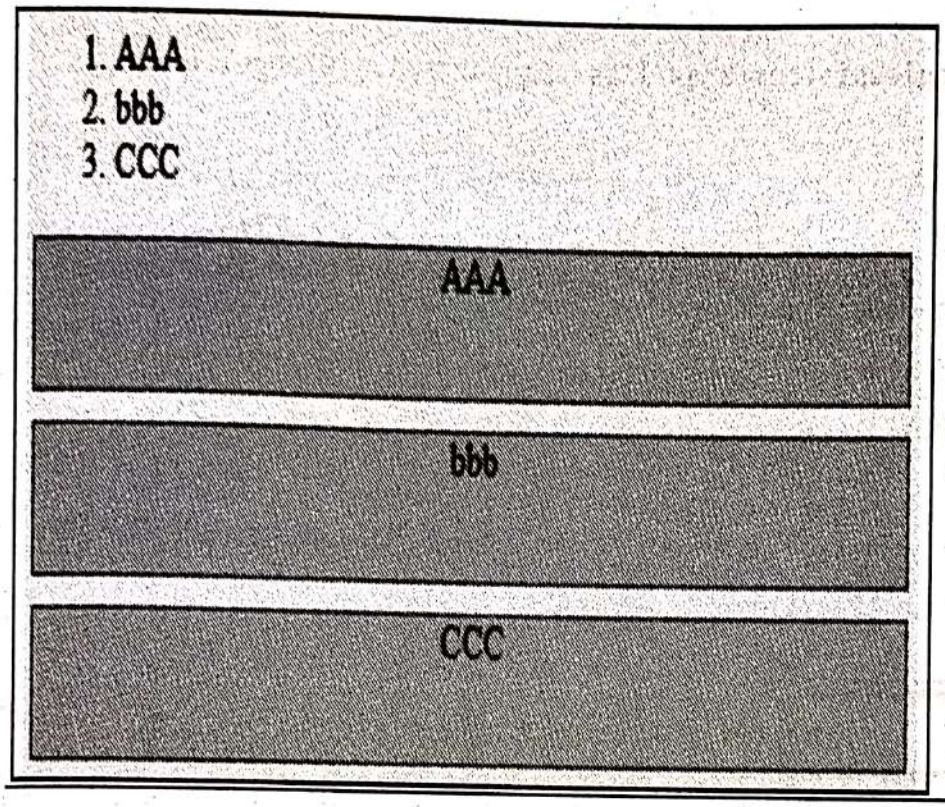
Variable	Details
$index	It contains the index of the element used for the traversal.
$first	It returns true if the repeated element is first in the iterator.
$middle	It returns true if the repeated element is between the first and last in the iterator.
$last	It returns true if the repeated element is last in the iterator.
$even	It returns true if the iterator position $index is even, otherwise return false.
$odd	It returns true if the iterator position $index is odd, otherwise return false.

Program 2.3: Program to demonstrate ng-repeat.

```
<!DOCTYPE html>
<html>
<head>
    <script
    src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.
    js"></script>
     <style>
         div {
             border: 1px solid red;
             width: 100%;
             height: 50px;
             display: block;
             margin-bottom: 10px;
             text-align:center;
             background-color: pink;
         }
    </style>
</head>
<body ng-app="" ng-init="students=['AAA','bbb','CCC']">
    <ol>
        <li ng-repeat="name in students">
            {{name}}
```

```
        </li>
    </ol>
    <div ng-repeat="name in students">
        {{name}}
    </div>
</body>
</html>
```

Output:



4. ng-model:

- When we bind a Model onto our HTML, we're tying Model data to elements.
- The ng-model directive is used for two-way data binding in AngularJS.
- It binds <input>, <select> or <textarea> elements to a specified property on the $scope object. So, the value of the element will be the value of a property and vice-versa.
- To initialize a new Model, if it doesn't exist yet or bind an existing Model then we simply bind to ng-model.
- ng-model is responsible for providing validation behavior (i.e. required, number, email, url).
- It is also responsible for keeping the state of the control (valid/invalid, dirty/pristine, touched/untouched, validation errors).

Syntax:

```
<input type="text" ng-model="message">
<p>Message is: {{ message }}</p>
```

Program 2.4: Program to demonstrate ng-model.

```
<!DOCTYPE html>
<html >
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
</head>
<body ng-app>
<input type="text" ng-model="name" />
<div>
Hello <h2>{{name}}</h2>
</div>
</body>
</html>
```

Output:

As you can see in the output as we are changing the text in input field the text is getting displayed on browser. So ng-model provides you two way binding.

5. **ng-click:**

- The ng-click directive allows us to specify custom behavior when an element is clicked.
- Angular will evaluate the expression(s) inside the ng-click for us and bind the relevant listeners.
- This makes our development much faster, imagine binding event listeners and call-backs to DOM elements manually, adding and removing them whilst the DOM is destroyed and recreated on the fly.

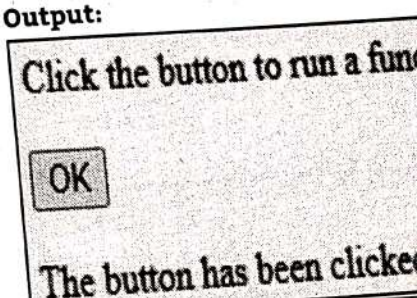
Syntax:

```
<input type="text" ng-model="main.mes
<a href=" ng-click="main.showMessage
```

Program 2.5: Program to demonstrate ng-cl

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/a
script>
<body ng-app="myApp">
    <div ng-controller="myCtrl">
      <p>Click the button to run
      <button ng-click="myFunc()"
      <p>The button has been cli
    </div>
    <script>
      angular.module('myApp', [
       .controller('myCtrl', ['$
            $scope.count = 0;
            $scope.myFunc = func
            $scope.count++;
            };
      }]);
    </script>
</body>
</html>
```

Output:



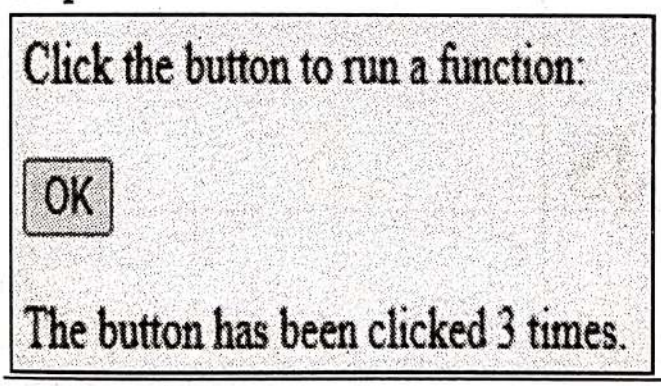
Syntax:

```
<input type="text" ng-model="main.message">
<a href=" ng-click="main.showMessage(main.message);">Show my message</a>
```

Program 2.5: Program to demonstrate ng-click.

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></
script>
<body ng-app="myApp">
    <div ng-controller="myCtrl">
      <p>Click the button to run a function:</p>
      <button ng-click="myFunc()">OK</button>
      <p>The button has been clicked {{count}} times.</p>
    </div>
    <script>
      angular.module('myApp', [])
      .controller('myCtrl', ['$scope', function($scope) {
          $scope.count = 0;
          $scope.myFunc = function() {
          $scope.count++;
          };
      }]);
    </script>
</body>
</html>
```

Output:



6. **ng-show/ng-hide**

- Using ng-show and ng-hide are often quite common to use within Angular, it's a fantastic way to show and hide data based on a $scope property's value.
- The ng-show and ng-hide directives in AngluarJS are used to show or hide the specified HTML element.
- If given expression in ng-show attribute is true then the HTML element will display otherwise it hide the HTML element.
- If the expression given in the ng-hide attribute is true than the HTML elements hide.
- ng-hide is also a predefined CSS class in AngularJS, and sets the element's display to none.
- There is no other difference apart from they do the opposite of each other.

Syntax:

```
<a href=" ng-click="showMenu = !showMenu"> menu!</a>
<ul ng-show="showMenu">
  <li>1</li>
  <li>2</li>
  <li>3</li>
</ul>
```

Program 2.6: Program to demonstrate ng-show.

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
<body ng-app="">
Show HTML element: <input type="checkbox" ng-model="myVar">
<div ng-show="myVar">
<h1>Welcome to Angular JS</h1>
<p>An Example of ng show</p>
</div>
</body>
</html>
```

Output:

7. ng-if:

- The ng-if directive removes or recreates a portion of the DOM tree based on an {expression}. If the expression assigned to ng-if evaluates to a false value then the element is removed from the DOM, otherwise a clone of the element is reinserted into the DOM.
- Unlike ng-show or ng-hide, which uses a class to toggle element's visibility on the page, ng-if actually destroys the DOM and the $scope it creates.
- If the element needs recreating due to value changes, Angular will create new $scope for that particular element.
- This also adds to improved performance, as removing elements via ng-if removes the $scope, which in turn lowers the $$watchers count inside Angular, speeding up further $digest cycles.

Syntax:

```
<element ng-if="expression"></element>
```

Program 2.7: Program to demonstrate ng-if.

```
<!DOCTYPE html>
<html >
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
</head>
<body ng-app>
<label>Click me:<input type="checkbox" ng-model="checked"
ng-init="checked=true"/>
</label><br/>
Show when checked:
<span ng-if="checked">
This is removed when the checkbox is unchecked.
</span>
</body>
</html>
```

Output:

Click me: ☑
Show when checked: This is removed when the checkbox is unchecked.

8. **ng-switch**

- It is typical same as switch case, but in the DOM, an ng-if is used.
- It provides AngularJS multiple chunks of HTML for it to swap in and out based on a single $scope value.
- This allows us to provide different HTML for different users.

Syntax:

```
<element ng-switch="expression">
  <element ng-switch-when="value"></element>
  <element ng-switch-when="value"></element>
  <element ng-switch-when="value"></element>
  <element ng-switch-default></element>
</element>
```

For example,

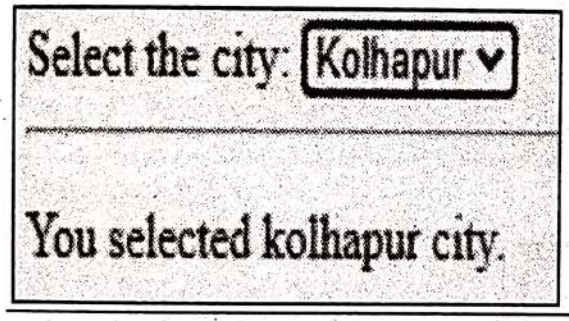
```
<div ng-switch on="main.user.access">
  <div ng-switch-when="admin">
    <!-- code for admins -->
  </div>
  <div ng-switch-when="user">
    <!-- code for users -->
  </div>
  <div ng-switch-when="author">
    <!-- code for authors -->
  </div>
</div>
```

Program 2.8: Program to demonstrate ng-switch.

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
<body ng-app="">
Select the city:
<select ng-model="myVar">
<option value="Pune">Pune
```

```
<option value="mumbai">Mumbai
<option value="Nagpur">Nagpur
<option value="Kolhapur">Kolhapur
</select>
<hr>
<div ng-switch="myVar">
<div ng-switch-when="Pune">
<p>You selected pune city.</p>
</div>
<div ng-switch-when="mumbai">
<p>You selected mumbai city.</p>
</div>
<div ng-switch-when="Nagpur">
<p>You selected Nagpur city.</p>
</div>
<div ng-switch-when="Kolhapur">
<p>You selected kolhapur city.</p>
</div>
<div ng-switch-default>
<h1>Hello</h1>
<p>Select any city please.</p>
</div>
</div>
</body>
</html>
```

Output:



9. ng-bind:

- The ng-bind Directive in AngularJS is used to bind/replace the text content of any particular HTML element with the value that is entered in the given expression.

- The value of specified HTML content updates whenever the value of the expression changes in ng-bind directive.
- The ng-bind directive binds the model property declared via $scope or ng-model directive or the result of an expression to the HTML element.
- It also updates an element if the value of an expression changes.

Syntax:

```
<element ng-bind="expression"> Contents... </element>
```

Where expression is used to specify the expression to be evaluated or the variable.

Program 2.9: Program to demonstrate ng-bind.

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.2/angular.min.js">
</script>
</head>
<body ng-app="app">
    <div ng-controller="controllerName">
        <label>Enter Data: <input type="text" ng-model="name" /> </label>
        <br />
        <b>Binded data:</b> <span ng-bind="name">
        </span>
    </div>
<script>
    var app = angular.module("app", []);
    app.controller('controllerName', ['$scope', function ($scope) {
        $scope.name = 'ABC';
    }]);
</script>
</body>
</html>
```

Output:

Enter Data: ABC

Binded data: ABC

10. ng-readonly

- The AnglarJS readonly directive sets the readonly attribute on the element; if it gets that the expression inside ng-readonly is true.
- It is only applied to input elements with specific types. The ng-readonly directive is necessary to enable to shift the values between true and false. In HTML, readonly attributes cannot be set to false.
- It is supported by <input> <textarea> elements.

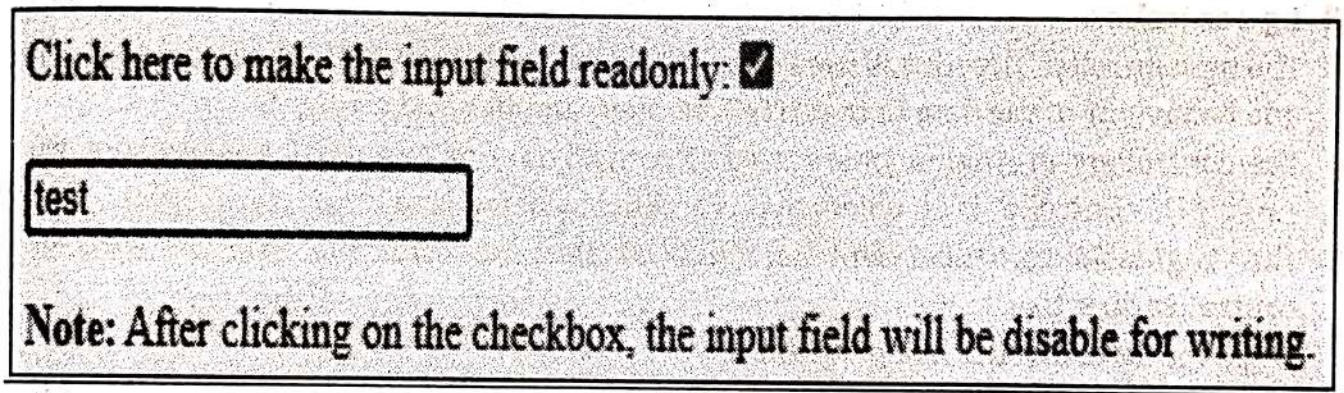
Syntax:

```
<input ng-readonly="expression"></input>
```

Program 2.10: Program to demonstrate ng-readonly.

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<body ng-app="">
Click here to make the input field readonly:<input type="checkbox" ng-
model="all"><br>
<br>
<input type="text" ng-readonly="all">
<p><strong>Note:</strong> After clicking on the checkbox, the input field
will be disable for writing.</p>
</body>
</html>
```

Output:



11. ng-disabled

- The ng-disabled Directive in AngularJS is used to enable or disable HTML elements.
- If the expression inside the ng-disabled attribute returns true then the form field will be disabled or vice versa. It is usually applied on form field (input, select, button, etc).

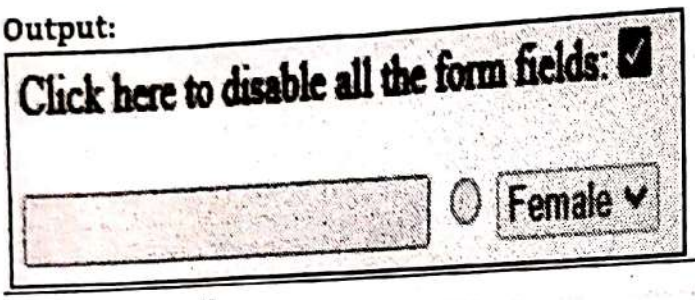
Syntax:

```
<element ng-disabled="expression"> Contents... </element>
```

Program 2.11: Program to demonstrate ng-disabled.

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js
</script>
<body ng-app="">
Click here to disable all the form fields:<input type="checkbox"
model="all"><br>
<br>
<input type="text" ng-disabled="all">
<input type="radio" ng-disabled="all">
<select ng-disabled="all">
<option>Female</option>
<option>Male</option>
</select>
</body>
</html>
```

Output:



12. ng-controller:

- The ng-controller directive is used to specify a controller in HTML element, which wi add behaviour of the data in that HTML element and its child elements.
- The controllers in AngularJS are used to control data of AngularJS applications. Th AngularJS controller is an object that is created by using JavaScript constructor.
- The AngularJS controller will handle DOM elements with the help of using ng controller directive.
- Steps for ng-controller in a program:
  1. Specify a controller using ng-controller.
  2. Create an App module.
  3. Create a controller
  4. Attach a property to $scope
  5. Use a property created inside a controller.

Syntax:

```
<element ng-controller="expression"></element>
```

Program 2.12: Program to demonstrate ng-controller. Adjust following program box and arrows.

```
<!DOCTYPE html>
<html>
<head>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js"></script>
</head>
<body ng-app="myNgApp">
  <div ng-controller="myController">
    {{message}}
  </div>
  <script>
    var ngApp = angular.module('myNgApp', []);

    ngApp.controller('myController', function ($scope) {
      $scope.message = ".....Welcome you all BBA-CA students...";
    });
  </script>
</body>
</html>
```

Step 1

Step 5

Step 2

Step 3

Step 4

Output:

.....Welcome you all BBA-CA students...

2.3 EXPRESSION AND DATA BINDING

2.3.1 AngularJS Expressions

- AngularJS binds data to HTML using Expressions.
- Expressions in AngularJS are used to bind application data to HTML.
- The expressions are resolved by Angular and the result is returned back to where the expression is written.
- The expressions in AngularJS are written in double braces: `{{ expression }}`. They behave similar to ng-bind directives: `ng-bind="expression"`.

- AngularJS will resolve the expression, and return the result exactly where the expression is written.
- AngularJS expressions are much like JavaScript expressions: They can contain literals, operators, and variables.

Syntax:

```
{{ expression }}
```

Example: `{{ 5 + 5 }}` or `{{ firstName + " " + lastName }}`

Program 2.13: Program to demonstrate expression.

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
<body>
<div ng-app>
<p>My first expression: {{ 1 + 6 }}</p>
</div>
</body>
</html>
```

Output:

My first expression: 7

The key difference between angular expressions and JavaScript expressions:

- Like JavaScript, Angular expressions are code snippets that are usually placed in binding such as

  `{{ expression }}`

- The key difference between the JavaScript expressions and Angular expressions is:
  - **Context:** In Angular, the expressions are evaluated against a scope object, while the JavaScript expressions are evaluated against the global window
  - **Forgiving:** In Angular expression, evaluation is forgiving to null and undefined; while in JavaScript undefined properties generate `TypeError` or `ReferenceError`.
  - **No Control Flow Statements:** Loops, conditionals or exceptions cannot be used in an Angular expression.
  - **Filters:** You can use filters to format data before displaying it.

2.3.2 AngularJS Data Binding

- Data binding in AngularJS is the synchronization
- Data binding is the connection between data or
  to the end user. For example, you can have a fiel
- Data binding is a very useful and powerful
  technologies. It acts as a bridge between
  application.
- Following are the two types of data binding
  1. One-way data
  2. Two-way data

1. One-Way Data Bi

- The One-way data b
  into an

2. Two-W

- Angular
- Two-way
  between t
- Two-way
  applica
  th

2.3.2 AngularJS Data Binding

- Data binding in AngularJS is the synchronization between the model and the view.
- Data binding is the connection between data or business rules and their presentation to the end user. For example, you can have a field for the make/model of a vehicle.
- Data binding is a very useful and powerful feature used in software development technologies. It acts as a bridge between the view and business logic of the application.
- Following are the two types of data binding in AngularJS:
  1. One-way data binding
  2. Two-way data binding

1. One-Way Data Binding:

- The One-way data binding is an approach where a value is taken from the data model and inserted into an HTML element. There is no way to update model from view.
- It is used in classical template systems. These systems bind data in only one direction.

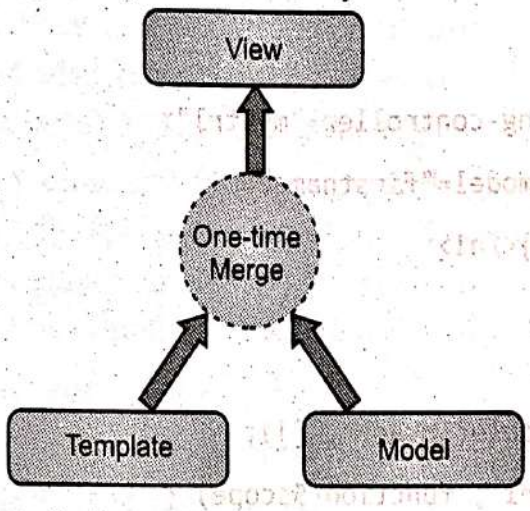


Fig. 2.1: One-way Data Binding

2. Two-Way Data Binding:

- AngularJS follows Two-way data binding model.
- Two-way data binding in AngularJS apps is the automatic synchronization of data between the model and view components.
- Two-way data binding lets you treat the model as the single source of truth in your application. The view is a projection of the model at all times. If the model is changed, the view reflects the change and vice versa.

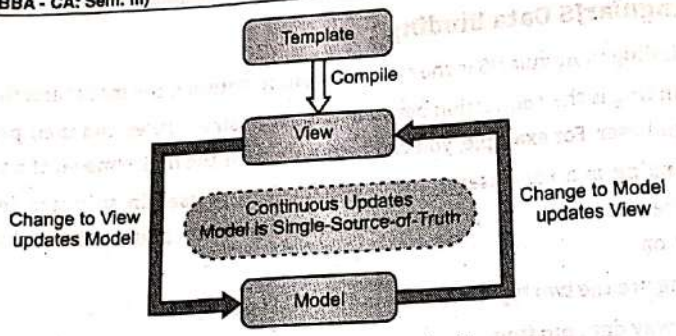


Fig. 2.2: Two-way Data Binding

Program 2.14: Write a program for data binding.

```
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
    Name: <input ng-model="firstname">
    <h1>{{firstname}}</h1>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstname = "Nirali";
    $scope.lastname = "Publication";
});
</script>
<p>Change the name inside the input field, and the model data will change
automatically, and therefore also the header will change its value.</p>
</body>
</html>
```


Output:

Name: Nirali

Nirali

Change the name inside the input field, and the model data will change autom
also the header will change its value.

In the above Program, the {{ firstname }} expression is a
expression. Data binding in AngularJS binds AngularJS expressi
{{ firstname }} is bound with ng-model="firstname".

2.4 WORKING WITH DIRECTIVES

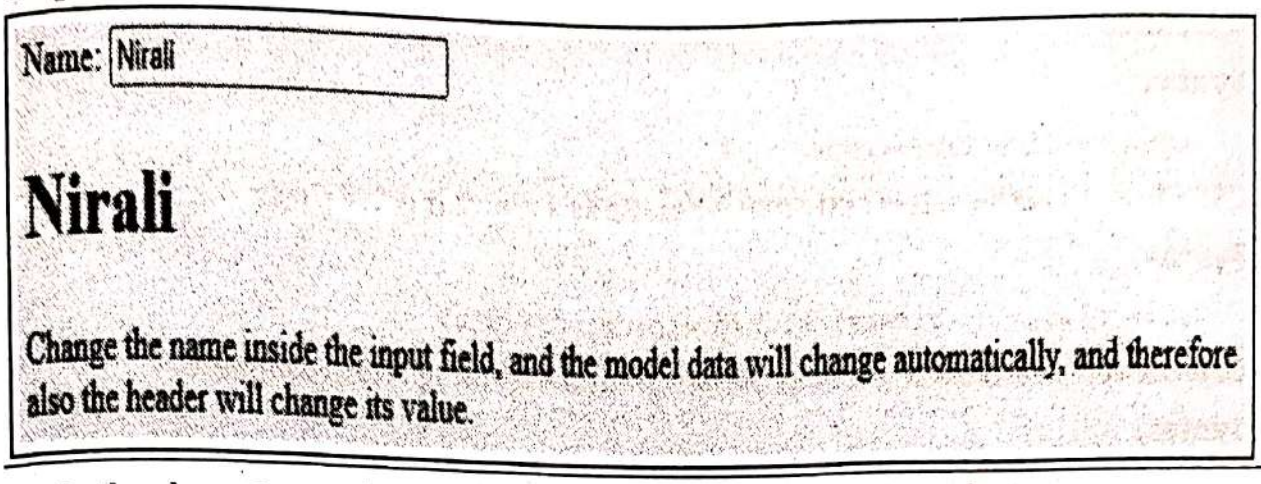
- Dire one of the mos ortant components of
are e attribute
- In other
- All th
directive c
element; ng-a
- These
manipulate the
When we are giving na
TestDirective

1. E
templat

Syntax:

```
<my-directive> </my
```

Output:



In the above Program, the {{ firstname }} expression is an AngularJS data binding expression. Data binding in AngularJS binds AngularJS expressions with AngularJS data. {{ firstname }} is bound with ng-model="firstname".

2.4 WORKING WITH DIRECTIVES

- Directives are one of the most important components of AngularJS application. They are extended HTML attributes.
- In other words, directives are something that introduces new syntax. They are markers on the DOM element which provides some special behaviour to DOM elements and tell AngularJS's HTML compiler to attach.
- There are many built-in directives such as ng-model, ng-repeat, ng-show, ng-bind etc. All these directives provide special behaviour to DOM elements. For example, ng-show directive conditionally shows an element, ng-click directive adds click events to the element; ng-app directive initializes an AngularJS application, etc.

2.4.1 Custom Directives

- AngularJS allows to create custom directives with which it becomes easier to encapsulate and simplify DOM manipulation.
- These directives extend the HTML functionality. Also new Directives can be created to manipulate the HTML behavior.
- When we are giving name to a directive, we must use a camel case name e.g. MyTestDirective, but when we invoking it, we must use '-' for separate name like My-Test-Directive.

Following are the ways to implement custom directives in AngularJS :

1. **Element directives:** It is activated when find a matching HTML element in the HTML template.

   Syntax:

   ```
   <my-directive> </my-directive>
   ```

2. **Attribute directives:** It is activated when finds a matching HTML attribute in th template.

   Syntax:

   ```
   <div my-directive></div>
   ```

3. **CSS Class directives:** It is activated when finds a matching CSS Class.

   Syntax:

   ```
   <div class="my-directive: expression; "></div>
   ```

4. **Comment directives:** It is enabled when finds a matching HTML comment.

   Syntax:

   ```
   < !-- Directive : my-directive expression; -->
   ```

Following are some commonly used properties of directive:

1. **restrict:** It specify how a directive is implemented in angular app. There are 4 restrict options:
   - **restrict : 'A'** – attribute (default one) | <div my-directive>
   - **restrict : 'C'** – Class |
   - **restrict : 'E'** – element | <my-directive></my-directive>
   - **restrict : 'M'** – Comment | <!– Directive : my-directive expression; –>

     You can also specify multiple restrict like as : restrict : 'EC'.
2. **scope:** Scope accesses data or methods inside template and link function. By default, the directives do not produce their own scope without explicitly set. Different types of scopes are able to define which are as follows:
   - **scope : true** – Get new scope.
   - **scope : false** – Use its parent scope.
   - **scope : {}** – Get new isolated scope that doesn't inherit from parent and exists on its own.
3. **template:** It specifies the HTML content that will be generated when directive is compiled.
4. **templateUrl:** It specifies the path of template that should be used by directives.

Program 2.15: Program to create custom directive.

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.j
s"></script>
<body ng-app="myApp">
```

```
<my-Test-Directive></my-Test-Directive>
<script>
var app = angular.module("myApp", []);
app.directive("myTestDirective", function() {
    return {
        template : "<h1>====Yes I have made my a directive====</h1>"
    };
});
</script>
</body>
</html>
```

Output:

==Yes I have made my a directive==

2.4.2 Functions of AngularJS Directive Life Cycle

- The directive lifecycle begins and ends within the AngularJS bootstrapping process, before the page is rendered.
- In a directive's life cycle, there are four distinct functions that can execute if they are defined. Each enables the developer to control and customize the directive at different points of the life cycle.

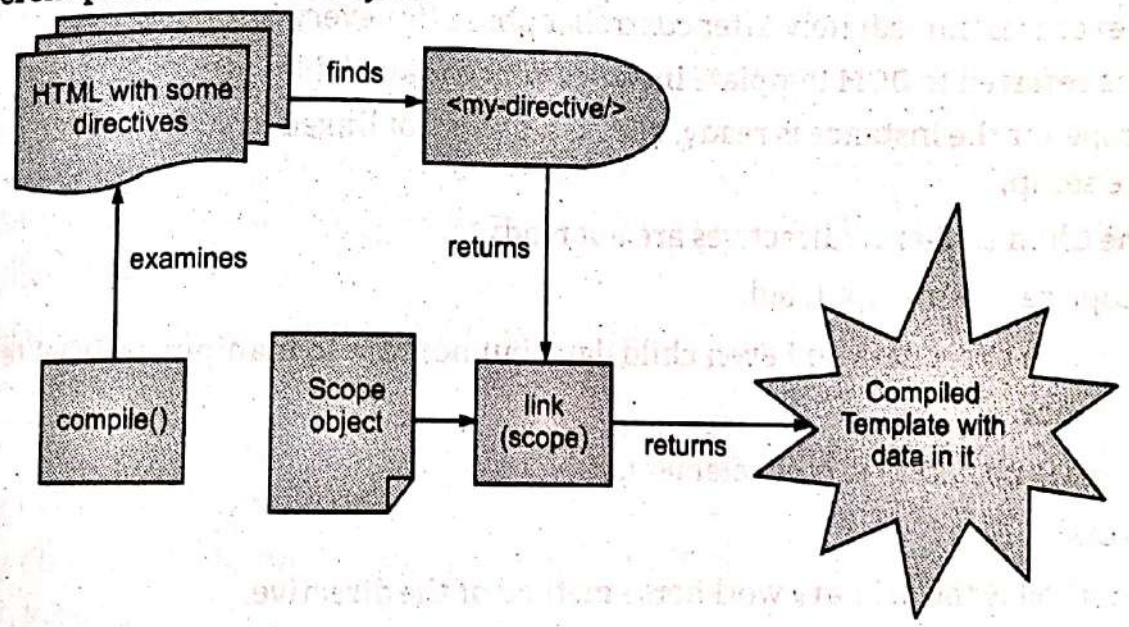


Fig. 2.3: AngularJS Directive Life Cycle

Following are 4 functions of AngularJS Directive Life Cycle:

1. **Compile:**
   - It allows the directive to manipulate the DOM before it is compiled and linked. And thus, allowing it to add/remove/change directives, as well as, add/remove/change other DOM elements.
   - It loads and traverses template DOM.
   - It executes only once.
   - Returns link function or object (containing pre, post (or link) etc.)
   - There is no scope present.
   - There is no instances/clones (of template) created yet.
   - It dose operations which can be shared among all instances/clones of template.
   - It can manipulate DOM of template.
   - It cannot play with data/events of clones.
   - There is no DOM of clones available.
2. **Controller:**
   It facilitates directive communication. Sibling and child directives can request the controller of their siblings and parents to communicate information.
   - It is the first, to execute for every instance/clone of template.
   - It creates scope (scope initialization) for the template instance.
   - It can manipulate data for template instance.
   - It is not recommended to access instance DOM.
3. **Pre-link:**
   It allows for private **$scope** manipulation before the post-link process begins.
   - It executes immediately after controller phase (for every instance).
   - It is referred to DOM template instance which is available.
   - Scope for the instance is ready, but Instance is not linked to scope yet (no bindings are setup).
   - The Child elements/directives are not ready.
   - Scope can be manipulated.
   - It is safe to set data and even child data, but not safe to manipulate DOM template instance.
   - There is no access to child elements.
4. **Post-link:**
   This method is the primary workhorse method of the directive.
   - It is the last phase, usually called as "linking".
   - It is referred to DOM template instance is available.

- In this the scope and instance are link
- Child elements/directives are ready (
- Scope can be manipulated.
- It is safe to attach event handlers an
- It is safe to manipulate DOM temp elements.

Summary

- Most of the directives in AngularJS s
- Directives are special attributes star
- ng-app : This directive starts an An
- ng-init : This directive initializes a
- ng-model : This directive defines t
- ng-repeat : This directive repeats
- ng new: Create a new Angular ap
- ng update: updates your apps &
- ng generate: Develop Angular f
- ng execute: Assassinate Angul
- ng build: Create an application
- ng lint: Run code delivery to a
- Element directive, arrtribute the different ways to implement custom
- Data binding is a very use technologies. It acts as application.
- There are two types of d Two-way data binding.
- The directive lifecycle be before the page is rende
- In a directive's life cycl link, post-link.

- In this the scope and instance are linked (and data bound).
- Child elements/directives are ready (and already linked).
- Scope can be manipulated.
- It is safe to attach event handlers and inspect child elements.
- It is safe to manipulate DOM template instance but not safe to set data for child elements.

Summary

- Most of the directives in AngularJS starts with ng- where ng stands for aNGular.
- Directives are special attributes starting with ng- prefix.
- 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 new: Create a new Angular app.
- ng update: updates your apps & its dependencies.
- ng generate: Develop Angular files.
- ng execute: Assassinate Angular app.
- ng build: Create an application for deployment.
- ng lint: Run code delivery to analyse code.
- Element directive, arrtribute directive, CSS class derictive and comment directive are the different

  ways to implement custom directives.
- Data binding is a very useful and powerful feature used in software development technologies. It acts as a bridge between the view and business logic of the application.
- There are two types of data binding. First is a One-way data binding and second is Two-way data binding.
- The directive lifecycle begins and ends within the AngularJS bootstrapping process, before the page is rendered.
- In a directive's life cycle, there are four distinct functions: compiler, controller, pre-link, post-link.

Check Your Understanding

1. Which of the following is true about ng-app directive?
   (a) ng-app directive defines and links an AngularJS application to HTML.
   (b) ng-app directive indicates the start of the application.
   (c) Both of the above.
   (d) None of the above.
2. AngularJS application expressions are pure JavaScript expressions.
   (a) true (b) false
3. Which of the following is true about ng-hide directive?
   (a) ng-hide directive can show a given control.
   (b) ng-hide directive can hide a given control.
   (c) Both of the above.
   (d) None of the above.
4. AngularJS expressions are written using.....
   (a) (expression) (b) {{expression}}
   (c) {{{expression}}} (d) [expression]
5. Which Angular directive is used to binds the value of HTML controls (input, select, textarea) to application data?
   (a) ng-cloak (b) ng-bind
   (c) ng-model (d) ng-view
6. Which Angular Directive is used to disable an Element?
   (a) ng-disabled (b) ng-hide
   (c) ng-false (d) ng-view
7. AngularJS directives are used in .....
   (a) Module (b) Controller
   (c) Service (d) View
8. Which of the following directive allows us to use a form in AngularJs?
   (a) Ng-include (b) Ng-form
   (c) Ng-directive (d) Ng-bind
9. Which of the followings are validation directives?
   (a) ng-required (b) ng-minlength
   (c) ng-pattern (d) All of the above

10. The ..... directive is used if you want to add or remove HTML elements from the DOM based on data in the model.
    (a) ng-switch
    (b) ng-model
    (c) ng-Disabled
    (d) ng-Cloak
11. Which of the following directive is used to initialize an angular app?
    (a) ng-model
    (b) ng-app
    (c) ng-controller
    (d) None of the above

ANSWER

1. (c)	2. (a)	3. (c)	4. (b)	5. (c)	6. (a)	7. (d)	8. (b)	9. (d)	10. (a)
11. (b)									

Practice Questions

Q. I Answer the following questions in short.

1. What is data binding in AngularJS?
2. Explain ng-click directives in AngularJS with example.
3. What does Directive mean in AngularJS?
4. Explain ng-controller directive in AngularJS.
5. Explain ng-if directive in AngularJS.
6. What is use of ng-disable directive in AngularJS?

Q. II Answer the following questions.

1. What is Angular Expression? Explain the key difference between angular expressions and JavaScript expressions.
2. What are directives? Name some of the most commonly used directives in AngularJS application.
3. What are the most common directives used in AngularJS applications?
4. Why are expressions used in AngularJS?
5. What are different variables used with the ng-repeat directive?
6. Explain working of directive with AngularJS.
7. Write a program to create custom directive in AngularJS.

Q. 3) Define the following terms

1. Directive
2. Expression
3. Data Binding
4. Two-way data binding

•••

3...

AngularJS Modules, Controller, View and Scope

Objectives...

- To understand AngularJS Modules and Controllers.
- To learn AngularJS View.
- To understand about the Scope hierarchy.

3.1 INTRODUCTION

- Modules are AngularJS's way of packaging relevant code under a single name.
- A module can define its own controllers, services and directives. These are functions and code that can be accessed throughout the module.
- Controllers in AngularJS are our workhorse, the JavaScript functions that perform or trigger the majority of our UI oriented work.
- An AngularJS controller is almost always directly linked to a view or HTML.
- The view is the UI that the user sees and interacts with. It is dynamic, and generated based on the current model of the application.

3.2 ANGULAR MODULES

- AngularJS supports modular approach.
- Module serves as a container of different parts of your app such as controllers, services, filters, directives, etc.
- Modules can be referenced by other modules through AngularJS's dependency injection mechanism.
- Modules are used to separate logic such as services, controllers, applications etc. from the code and maintain the code clean.

Why to use Modules?

- Most applications have a main method that represents and wires together the different parts of the application.
- Module is starting point of AngularJS application.
- Angular apps don't have main method. But in AngularJS, the declarative process is easy to understand and one can package code as reusable modules.
- ng-app that have empty module make its controller function global.
- Controller function will become global if we don't register it with the module then there may be chances of overriding data.
- To avoid this, we have register controller function with the module. So that can solve global value problem.

3.2.1 Creating a Module

- Creating modules in AngularJS is very simple, you have to just call angular.module() method.
- Module is collection of controllers, directives, filters, services and other configuration information.
- Angular.Module() is the gateway into the module API. It is used to register, create and retrieve previously created AngularJS Modules.
- Injecting a module as a dependency of another module. For example,

```
angular.module('app', ['app.auth','app.dashboard']);
```

Declaration of a module:

Syntax:

```
angular.module(name, [requires], [configFn]);
```

Where,

name: Name of the module.

requires: It is an optional parameter, if mentioned the new module gets created. If not, an existing module is being retrieved.

configFn: It is an optional parameter, execute a function on module load.

Example:

```
var app = angular.module('myApp', []);
// Empty array is list of modules myApp is depends on.
// if there are any required dependencies, then you can add in module,
   Like ['ngAnimate'].
app.controller('myController', function()
{
   // write your business logic here
});
```

3.2.2 Modules Lifecycle

- AngularJS splits module's lifecycle into two phases, which are: configuration phase and run phase.

1. **The configuration phase:**
   - It gets executed during provider and configuration phase.
   - It is the phase where all the recipes are collected and configured.
   - The configuration phase allows us to do the last moment tweaks to the object's creation formula.
   - AngularJS can have multiple configure blocks.
2. **The run phase:**
   - It gets executed after the injector is created and it is used to start the application.
   - It allows us to register any work that should be executed upon the application's bootstrap.
   - AngularJS can have multiple run blocks.

3.2.3 Advantages of Modules

- Module uses for packaging the code in reusable modules.
- The process of declaration is easy to understand.
- Modules can be loaded in random manner.
- It is an easily testable component.
- It is an easily maintainable component.
- Using a module, you can easily organize an application.

Program 3.1: Program to demonstrate use of angular module.

```
<html ng-app="myApp">
<head><title>Program for Angular Module</title></head>
<body>
Hello {{1 + 1}}nd time AngularJS
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.js">
</script>
<script type="text/javascript">
angular.module('myApp', []);
</script>
</body>
</html>
```

Output:

Hello 2nd time AngularJS

3.3 ANGULAR CONTROLLER

- AngularJS application depends on controllers that control the data flow in the application.
- The main part of the controller is to build a model for view, to display various details on page. Here, model is nothing but data. A controller may call a web Server which retrieves a data from a database.
- Controllers are JavaScript objects that contain attributes/properties and are defined using ng-controller directive.
- Controllers maintain the application logic and provide logic to the view. Views are the HTML pages i.e., where a user can see the output.
- Controllers in AngularJS are java script functions that perform or trigger majority of our UI oriented work.
- Some of the common responsibilities of a controller in an AngularJS module includes:
  1. Fetching the right data from the server for current UI.
  2. Deciding which part of the data to show to users.
  3. Presentation logic, such as how to display elements, which parts of UI to show, how to style them etc.
  4. User interactions, like what happens when user clicks on something or how text input should be validated.

How to create a Controller in AngularJS ?

```
var myController=function($scope)
$scope.message="Hello";
```

- We are creating an anonymous function here, which is *function($scope)* and we are assigning it to a variable *myController*. Here, we are passing a parameter called *$scope*.
- *$scope* is nothing but an angular object that is passed to the controller function by the angular framework, automatically. We attached the model to this *$scope* object which will be then available in the respective views.
- Now, in the view, we are using data binding expression to display details. Here, we are passing *message* property to this *$scope* object and it's storing a string.

Program 3.2: Program to show how to create controller for myApp module.

```
<!DOCTYPE html>
<html ng-app="myApp">
<head><title>AngularJS Program for controller</title></head>
<body ng-controller="MyCtrl">
<h1>I have created my First Controller</h1>
```

```
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.js">
</script>
<script type="text/javascript">
angular.module('myApp', [])
.controller('MyCtrl', [function() {
// Controller-specific code goes here
console.log(' MyCtrl controller has been created');
}]);
</script>
</body>
</html>
```

Output:

I have created my First Controller

- Here we have defined a controller using the controller function that is exposed on AngularJS Module. The controller function takes names of controller as first argument.
- Here MyCtrl is the name of the controller. The second argument is the actual controller definition, saying what is does.
- We have also introduce a new directive, ng-controller this is used to tell AngularJS to go to instantiate an instance of controller with the given name and attach it to DOM element. In our program MyCtrl will print console.log() statement.

3.3.1 Modules and Controllers

- In AngularJS, the pattern used for developing modern day web applications is of creating multiple modules and controllers to logically separate multiple levels of functionality.
- Normally modules will be stored in separate JavaScript files, which would be different from the main application file.
- Now for our AngularJS Application with a controller, we will create a HelloWorld Application where Hello World Message from HTML to controller and get and display it from the controller.

Program 3.3: Program to create a HelloWorld Application where Hello World Message from HTML to controller and get and display it from the controller.

```
<!DOCTYPE html>
<html ng-app="myApp">
<head><title>Demo App</title></head>
<body ng-controller="MyCtrl as ctrl">
{{ctrl.helloMsg}} AngularJS@@@
<br/>
{{ctrl.goodbyeMsg}} I am AngularJS
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.js">
</script>
<script type="text/javascript">
angular.module('myApp', [])
.controller('MyCtrl', [function() {
this.helloMsg = 'Hello!!!';
var goodbyeMsg = 'Goodbye ';
}]);
</script>
</body>
</html>
```

Output:

Hello!!! AngularJS@@@
I am AngularJS

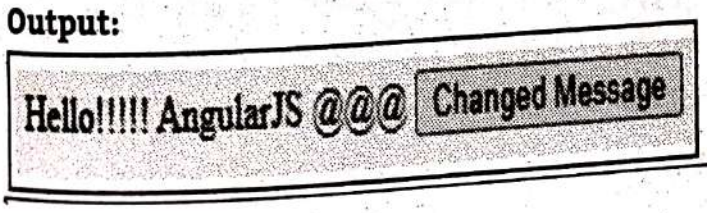
- Here we have only seen Hello AngularJS not Goodbye AngularJS in the UI.
- We have defined a module MyApp as we have seen before.
- We have created a Controller called MyCtrl using the controller function on the module.
- We have defined a message hellomsg on controller instance (using this). And the variable goodbyemsg as local inner variable in the controller's instance using var keyword.
- We have used the controller in UI using another directive ng-controller. This directive allows us to associate instance of controller with UI element (with body tag).
- We also gave particular instance of Myctrl a name when we used it in ng-controller, we have called it as ctrl. It is known as controllerAs syntax in angular JS.

- Then we have referred to the helloMsg and goodbyeMsg variables from the controller in HTML using double curly notations {{ }}.
- It is obvious that variables which are defined with this keyword in the controller are accessible from the HTML, but local, inner variables are not accessible.

Program 3.4: Sample program to display and change the message by using controller and module.

```
<!DOCTYPE html>
<html ng-app="myApp">
<head><title>Demo App</title></head>
<body ng-controller="MyCtrl as ctrl">
{{ctrl.message}} AngularJS @@@
<button ng-click ="ctrl.changeMsg()">
Changed Message
</button>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.js">
</script>
<script type="text/javascript">
angular.module('myApp', [])
.controller('MyCtrl', [function() {
var self=this;
self.message='Hello!!!!!';
self.changeMsg=function(){
self.message='Goodbye';
};
}]);
</script>
</body>
</html>
```

Output:



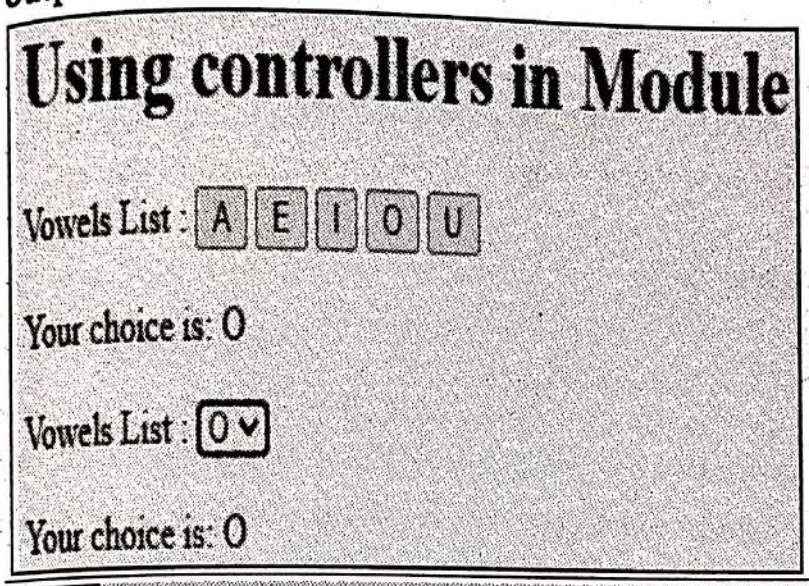
- As you can see in the previous example *goodbye* message was not displayed, but in this example, we have taken a button with label *Changed Message* there is built in directive on ng-click to which we have passes a function as argument. The ng-click directive evaluates any expression passed to it when button is clicked.
- The changeMsg() function in the controller sets value of message to Goodbye.

Program 3.5: Program to show controllers in Module.

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
</head>
<body ng-app="DemoApp">
<h1>
Using controllers in Module
</h1>
<div ng-app="DemoApp" ng-controller="DemoController">
Vowels List : <button ng-click="ch('A')">A</button>
<button ng-click="ch('E')">E</button>
<button ng-click="ch('I')">I</button>
<button ng-click="ch('O')">O</button>
<button ng-click="ch('U')">U</button>
<p>{{ choice }}</p>
Vowels List :
<select ng-options="option for option in list" ng-model="mychoice" ng-
change="c()">
</select>
<p>{{ choice }}</p>
</div>
<script>
var app = angular.module('DemoApp', []);
app.controller('DemoController', function ($scope) {
$scope.list = ['A', 'E', 'I', 'O', 'U'];
$scope.choice = 'Your choice is: TEST';
```

```
$scope.ch = function (choice) {
$scope.choice = "Your choice is: " + choice;
};
$scope.c = function () {
$scope.choice = "Your choice is: " + $scope.mychoice;
};
});
</script>
</body>
</html>
```

Output:



3.4 ANGULAR VIEW

- The view is the UI that the user sees and interacts with. It is dynamic and generated based on the current model of the application.
- Views are same as virtual DOM.
- AngularJS is also known as a single page application where, it means multiple views are displayed on a single page without reloading the page.
- The new page gets displayed on the current page itself. For this purpose, AngularJS has provided ng-view, ng-template directives and $routeProvider service.
- ng-view is a directive that complements the $route service by including the rendered template of the current route into the main layout file. Every time the current route changes, the included view changes with it according to the configuration of the $route service.

Syntax:

```
<div ng-app="AngularngviewApp">
<ng-view></ng-view>
</div>
```

Program 3.6: Program to demonstrate ng-view.

```
<!DOCTYPE html>
<html>
<head>
<title></title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angul
ar.js">
</script>
</head>
<body ng-app="myApp">
<div ng-controller="AppController">
    <h1>I have completed AngularJS ngView example</h1>
    <a href="page1">PAGE1</a><a href="page2">PAGE2</a>
    <br/><br/>
    <div class="ViewContainer">
     <div ng-view></div>
    </div>
</div>
</body>
</html>
```

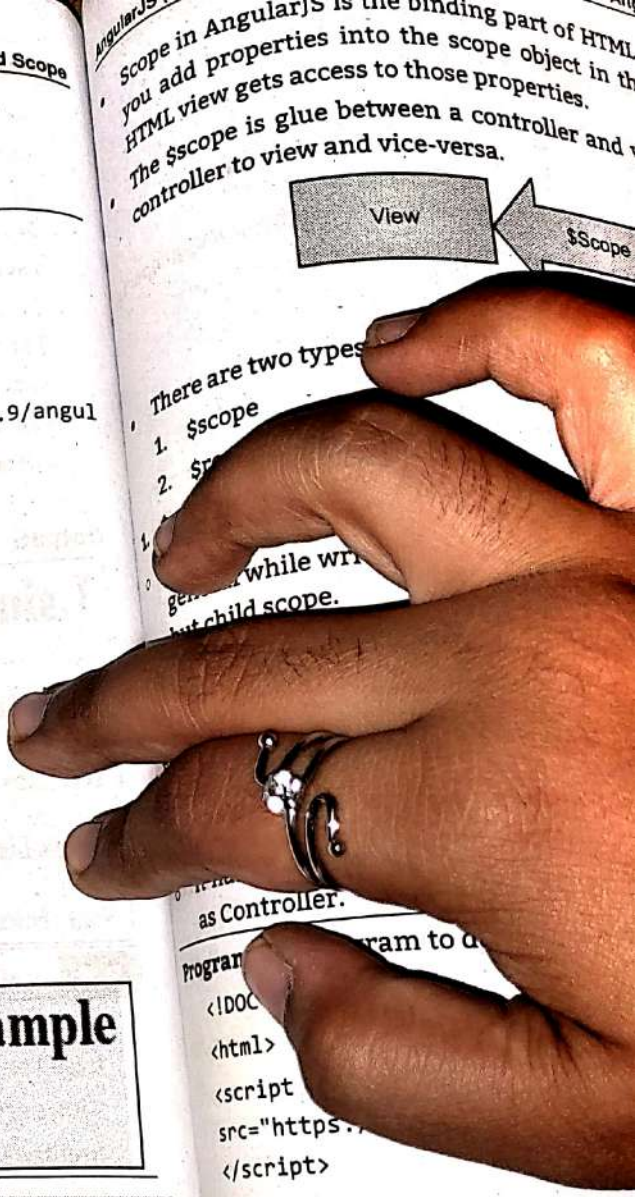
Output:

I have completed AngularJS ngView example

PAGE1 PAGE2

3.5 SCOPE HIERARCHY

- It is built-in object contains application data and methods.
- The scope is an object that acts as a shared context between the view and the controller that allows these layers to exchange information related to the application model. Both sides are kept synchronized along the way through a mechanism called two-way data binding.



- Scope in AngularJS is the binding part of HTML view and JavaScript controller. When you add properties into the scope object in the JavaScript controller, only then the HTML view gets access to those properties.
- The $scope is glue between a controller and view (HTML). It transfers data from the controller to view and vice-versa.

Fig. 3.1: Scope Concept

- There are two types of Scope in AngulerJS.
  1. $scope
  2. $rootscope

1. $scope:

- When we have controller then there would be child scope under the $rootScope. In general while writing controller we pass $scope object as an argument. This is nothing but child scope.

Syntax:

```
$scope
```

Features in Scope :

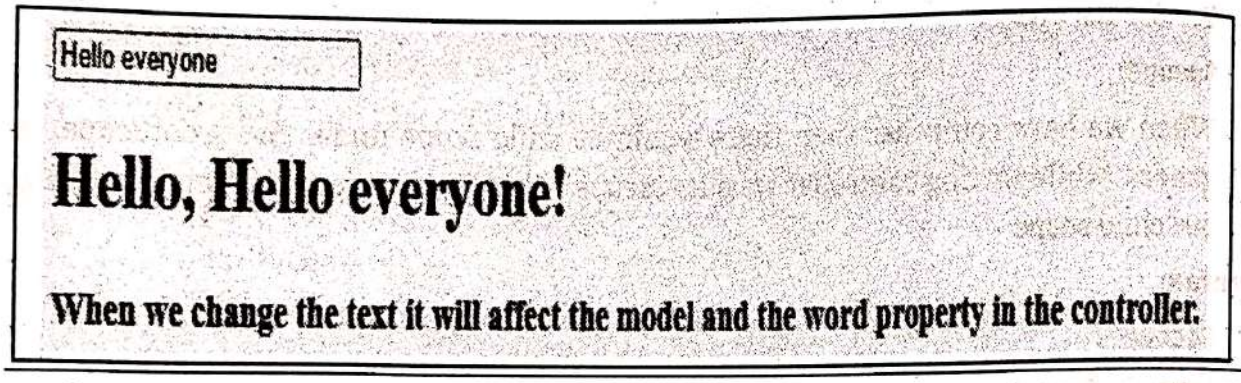
- It has the HTML view.
- It has the data which is available for the current view known as Model.
- It has the JavaScript function that makes/changes/removes/controls the data known as Controller.

Program 3.7: Program to demonstrate scope.

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
<body>
<div ng-app="MyApp" ng-controller="Mycontrol">
<input ng-model="text">
<h1>Hello, {{text}}!</h1>
</div>
<script>
```

```
var app = angular.module('MyApp', []);
app.controller('Mycontrol', function ($scope) {
$scope.text = "Hello everyone";
});
</script>
<h3>When we change the text it will affect the model and the word property
in the
controller.</h3>
</body>
</html>
```

Output:



2. $rootScope:

- If the variable has same name in the both rootscope and current scope then the controller or the application will use the current scope.
- It has a single $rootScope. All the other $scope objects are child objects of the root scope.
- The properties and methods attached to $rootScope will be available to all the controllers.
- Scopes provide separation between the model and the view, via a mechanism for watching the model for changes.
- To use $rootscope we need to inject into the controller.

Syntax:

```
$rootScope
```

Program 3.8: Program to demonstrate rootScope.

```
<!DOCTYPE html>
<html>
<head>
<script src=
```

```
"https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
</head>
<body ng-app="sco">
   <h2>Archana and Gajanan are {{relation}}</h2>
   <div ng-controller="control">
   <h2>Rekha and Seema {{relation}}</h2>
   </div>
   <script>
      var rs = angular.module('sco', []);
      rs.run(function($rootScope) {
      $rootScope.relation = 'Friends';
      });
      rs.controller('control', function($scope) {
      $scope.relation = "Sisters";
      });
   </script>
</body>
</html>
```

Output:

Archana and Gajanan are Friends

Rekha and Seema Sisters

Program 3.9: Program to create different scope.

```
<!DOCTYPE html>
<html>
<head>
<title>
Cities in Maharashtra
</title>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
```

```
</script>
</head>
<body>
<div ng-app="myApp" ng-controller="Mycontrol">
<ul>
<li ng-repeat="name in names">{{name}}</li>
</ul>
</div>
<script>
var cities = angular.module('myApp', []);
cities.controller('Mycontrol', function($scope) {
$scope.names = ["Pune", "Mumbai", "Aurangabad","Kolhapur" ];
});
</script>
</body>
</html>
```

Output:

- Pune
- Mumbai
- Aurangabad
- Kolhapur

Summary

- Modules are used to separate logic such as services, controllers, applications etc. from the code and maintain the code clean.
- The angular.module returns instance of a module and a controller can be defined directly on that instance via the controller function.
- The controller is used to build a model for view and to display various details on page.
- In AngularJS, $scope is used to achieve dependency injection and scope is used for linking between View (i.e. HTML) and Controller (i.e. JS).
- We can create a nested controller in AngularJS.

- A module can define its own controllers, services and directives. These are functions and code that can be accessed throughout the module.
- Controllers in AngularJS are our workhorse, the JavaScript functions that perform or trigger the majority of our UI oriented work.
- The view is the UI that the user sees and interacts with. It is dynamic, and generated based on the current model of the application.
- Module is starting point of AngularJS application.
- Angular apps don't have main method. But in AngularJS, the declarative process is easy to understand and one can package code as reusable modules.
- Angular.Module() is the gateway into the module API. It is used to register, create and retrieve previously created AngularJS Modules.

Check Your Understanding

1. What are Angular Controllers are responsible for .....
   (a) Controlling the data (b) Displaying the data
2. Which one of following is correct syntax for creating a module in AngularJS?
   (a) var myModule= angular.module(); (b) var myModule= new Module();
   (c) angular.module("app", []);
3. How do you share data between controller and view?
   (a) Using model (b) Using services
   (c) Using factory (d) Using $scope
4. angular.module is primarily used to create application module.
   (a) true (b) false
5. AngularJS controllers control ..... of AngularJS applications.
   (a) the data (b) the function
   (c) the variable
6. Scope contains the .....
   (a) model data (b) JavaScript Objects
   (c) variables
7. A controller is a JavaScript object containing.....
   (a) Attributes (b) Properties
   (c) Functions (d) All of the above

8. Which is the correct syntax of creating AngularJS controller?

(a)
```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
});
```

(b)
```
var app=angular.app('myApp', []);
app.controller('myCtrl', function($scope) {
});
```

(c)
```
var app = angular.module('myApp', []);
app.controller(function($scope) {
});
```

ANSWERS

1. (a)	2. (c)	3. (b)	4. (a)	5. (a)	6. (a)	7. (d)	8. (a)

Practice Questions

Q. I Answer the following questions in short.

1. What are the controllers in AngularJS?
2. What are the Angular Modules?
3. What is the difference between $scope and scope?
4. How to create a Controller in AngularJS?
5. What is scope hierarchy in AngularJS?

Q. II Answer the following questions.

1. How does angular.module work?
2. Write a short note on scope hierarchy.
3. Explain controller in detail.
4. Explain module life cycle.
5. Explain module in detail.
6. Write a short note on View.

Q. III Define the following terms.

1. Model
2. View
3. Controller
4. Scope hierarchy

4...

Filter, Forms and Ajax Filters

Objectives...

- To understand built in filters of AngularJS.
- To learn date, lowercase, uppercase, currency filters.
- To understand how to create custom filters and order by filters.
- To study angular forms, validations.
- To learn model binding, form controller, Using CSS classes, form events, custom model update triggers, custom validation.
- To study $http service.
- To study Ajax implementation using $http.

4.1 INTRODUCTION

- The AngularJS Filters are used to filter or format the data according to the user requirements.
- Filters can be used within controllers, services and directives. They can used to perform many types of transformations, but generally it is used to sort or format data model.
- Several filters are provided by Angular like number, currency, date. Even we can create custom filters in AngularJS.
- Filters can be used anywhere within the application, without being attached to any specific data or controller. Filters manipulate the data as it moves from the scope to the directive but the source data remains unchanged.
- A Form is a set of controls for the purpose of grouping related controls together that means it is a collection of input controls. Controls like Input, Select, TextArea, Checkbox, Radio buttons are the ways for a user to enter data.
- AngularJS gives us best solution for building full scale single page web applications. AngularJS provides us all the required features to create simple forms.

- In this chapter we will cover AngularJS filters, Forms, and $http Service.

4.2 BUILT-IN FILTERS

- Angular filters are used to modify the data and formatting the data. We can modify data according to our requirements. With the pipe(|) character we can use the filters. There are built-in filters and custom filters which we will cover in this chapter.

Syntax:

```
{{expression | name_of_filter}}
```

Example:

```
{{firstname | uppercase}}
```

- In this example name is the expression and uppercase is built in filter used to convert name into uppercase.

4.2.1 Lowercase and Uppercase Filters

1. Lowercase: Lowercase filter accept the input from user and convert it into lowercase.

Syntax:

```
{{expression | lowercase}}
```

2. Uppercase: Uppercase filter accept the input from user and convert into uppercase.

Syntax:

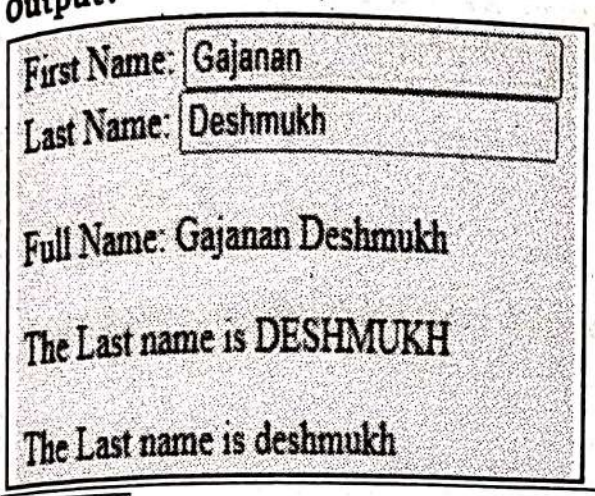
```
{{expression | uppercase}}
```

Program 4.1: Program for lowercase and uppercase filter.

```
<!DOCTYPE html>
<head>
   <script
   src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.m
   in.js">
   </script>
   <script>
      var app = angular.module('myApp', []);
      app.controller('myCtrl', function($scope) {
      $scope.firstName = "Gajanan";
      $scope.lastName = "Deshmukh";
      });
   </script>
```

```
</head>
<body>
   <div ng-app="myApp" ng-controller="myCtrl">
        First Name: <input type="text" ng-model="firstName"><br>
        Last Name: <input type="text" ng-model="lastName"><br>
        <br>
        Full Name: {{firstName + " " + lastName}}
        <p>The Last name is {{lastName|uppercase}}</p>
        <p>The Last name is {{lastName|lowercase}}</p>
   </div>
</body>
</html>
```

Output:



4.2.2 Date Filter

- Date filter is used to convert a Date into a specified format. When the Date format is not specified, the default Date format is 'MMM d, yyyy'.

Syntax:

```
{{expression | date: format: timezone }}
```

Following are the formats used for Date Filter:

- **'yyyy'**: Define 4 digit representation of year (e.g. output is 2020)
- **'yy'**: Define 2 digit representation of year (e.g. output is 20)
- **'y'**: Define 1 digit representation of year, (e.g. output is 2020)
- **'MMMM'**: Define Month in year (e.g. output is January-December)
- **'MMM'**: Define Month in year (e.g. output is Jan-Dec)
- **'MM'**: Define Month in year (e.g. output is 01-12)
- **'dd'**: Define Day in month, padded (e.g. output is 01-31)
- **'d'**: Define Day in month (e.g. output is 1-31)

- **'HH':** Define Hour in day (e.g. output is 00-23)
- **'H':** Define Hour in day (e.g. output is 0-23)
- **'mm':** Define Minute in hour (e.g. output is 00-59)
- **'m':** Define Minute in hour (e.g. output is 0-59)
- **'ss':** Define Second in minute (e.g. output is 00-59)
- **'s':** Define Second in minute (e.g. output is 0-59)
- **'a':** Define AM/PM marker (e.g. output is AM/PM)

Following are some localized formats:

- **'medium':** It is equivalent to *'MMM d, y h:mm:ss a'* (e.g. 5, 2020 12:08:10 PM)
- **'short':** It is equivalent to *'M/d/yy h:mm a'* (e.g. 4/5/20 11:10 PM)
- **'fullDate':** It is equivalent to *'EEEE, MMMM d, y'* (e.g. Monday, October 5, 2019)
- **'longDate':** It is equivalent to *'MMMM d, y'* (e.g. September 3, 2010)
- **'mediumDate':** It is equivalent to *'MMM d, y'* (e.g. Sep 3, 2010)
- **'shortDate':** It is equivalent to *'M/d/yy'* (e.g. 2/5/12)
- **'mediumTime':** It is equivalent to *'h:mm:ss a'* (e.g. 12:05:08 PM)
- **'shortTime':** It is equivalent to *'h:mm a'* (e.g. 12:05 PM)

Program 4.2: Program for Date filter.

```
<!DOCTYPE html>
<html>
<head>
    <script
    src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.m
    in.js"></script>
</head>
<body ng-app>
<div>
    Default date: {{1288323623006| date}} <br />
    Short date: {{1288323623006| date:'short'}} <br />
    Long date: {{1288323623006| date:'longDate'}} <br />
    Year: {{1288323623006 | date:'yyyy'}} <br />
    {{1288323623006 | date:'MM/dd/yyyy @ h:mma'}}
</div>
</body>
</html>
```

Output:

Default date: Oct 29, 2010
Short date: 10/29/10 9:10 AM
Long date: October 29, 2010
Year: 2010
10/29/2010 @ 9:10AM

4.2.3 Currency Filter

- It formats a number into currency for example ($1234256.78). When no currency sign is provided, the local currency is used. We can display the currency which is most commonly used in countries.

Syntax:

```
{{expression| currency: symbol: fraction size}}
```

- In the following program, we are binding the expression using ng-model. So the price which we will enter will be displayed with currency on browser.

Program 4.3: Program for Currency filter.

```
<!DOCTYPE html>
<html>
<head>
   <script
   src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.m
   in.js">
   </script>
</head>
<body ng-app >
   Enter Amount:<input type="number" ng-model= "price"><br>
   The amount is in Dollars: {{price | currency }}
   <br>
   The amount in Indian Rupees:{{price| currency:'₹' }}
   <br>
   The amount in Euro:{{price| currency:'€' }}
   <br>
   The amount in Pounds:{{price| currency:'£'}}
</body>
</html>
```

Output:

Enter Amount: 2000
The amount is in Dollars: $2,000.00
The amount in Indian Rupees:₹2,000.00
The amount in Euro:€2,000.00
The amount in Pounds:£2,000.00

4.2.4 Number Filter

- In AngularJS, number filter is used to format the number and convert it as string or text. By using number filter in AngularJS, we can show number with decimal values and we define limit to show number of decimal values.

  Syntax:

  ```
  {{number_expression | number:fractionsize}}
  ```

- For example, If we give expression like {{5000 | number :2}}. It will format our number and return result as 5,000.00.

Program 4.4: Program for Number filter.

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.j
s">
   </script>
</head>
<body ng-app >
   Enter Amount:<input type="number" ng-model= "price"><br>
   Number expression:{{price | number}}<br />
   Number with two Decimal Value: {{price | number:2}} <br />
   Number with four Decimal Value: {{price | number:4}} <br />
   Number with Zero Decimals:{{price | number:0}}
</body>
</html>
```

Output:

Enter Amount: 1230.2
Number expression:1,230.2
Number with two Decimal Value: 1,230.20
Number with four Decimal Value: 1,230.2000
Number with Zero Decimals:1,230

4.2.5 Filter in Filter

- Angular filter is used to filter the array of object elements and returned the filter elements. This filter selects a subset (a smaller array containing elements that meet the filter criteria) of an array from the original array.

Syntax:

```
{{ arrayexpression | filter : expression : comparator : anyPropertyKey}}
```

Where,

arrayexpression: The source array on which the filter will be applied.

expression: It is used to select the items from the array, after the filter conditions are met.

comparator: It is used to determining the value by comparing the expected value from the filter expression, and the actual value from the object array.

anyPropertyKey: It is a special property that is used to match the value against given property. Its default value is $.

Program 4.5: Program for Filter in Filter.

```
<!DOCTYPE html>
<html>
<head>
   <script
   src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.m
   in.js"></script>
   <script>
   angular.module('myApp',[]).controller('userCtrl', function($scope) {
         $scope.names = [
           'Suresh', 'Ramesh', 'Sachin', 'Santosh', 'Amol',
           'Gaurav', 'Samarth', 'Vikas', 'Deepak'
         ];
     });
   </script>
</head>
<body>
   <div ng-app="myApp" ng-controller="userCtrl">
       <ol>
       <li ng-repeat="x in names | filter : 'e'">{{ x }}</li>
       </ol>
   </div>
</body>
</html>
```

Output:

1. Suresh
2. Ramesh
3. Deepak

4.2.6 OrderBy Filter

- OrderBy filter allows us to sort an array with given expression. By default numbe are sorted numerically and characters are sorted alphabetically.

Syntax:

```
{{ array | orderBy : expression : reverse }}
```

Program 4.6: Program for OrderBy filter.

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
<body>
   <div ng-app="myApp" ng-controller="studentCtrl">
          <ul>
          <li ng-repeat="studentname in students | orderBy : 'name'">
          {{studentname.name + ", " + studentname.city}}
          </li>
          </ul>
   </div>
   <script>
          var app = angular.module('myApp', []);
          app.controller('studentCtrl', function($scope) {
          $scope.students = [
          {"name" : "Suresh", "city" : "Pune"},
          {"name" : "Ramesh ", "city" : "Mumbai"},
          {"name" : "Amol", "city" : "Nagpur"},
          {"name" : "Sunil", "city" : "Kolhapur"},
          ];
          });
   </script>
   <p>The array student items are arranged by "name".</p>
</body>
</html>
```

Output:

- Amol, Nagpur
- Ramesh , Mumbai
- Sunil, Kolhapur
- Suresh, Pune

The array student items are arranged by "name".

- In above program, in script tag we have created an array called student with name and city and using ng-repeat and by applying order by filter on name we are displaying all the names of students alphabetically. Note that if we write name then all the names will be displayed in descending order. Similarly we can apply filter on any field of an array.

4.2.7 Custom Filter

- In AngularJS we can create custom filters. AngularJS gives us a simple API to create a custom filter. As we use app.controller() to create controllers and app.module() to create modules. In exactly the same way, AngularJS has given us the angular.filter API to create a custom filter in AngularJS. A custom filter can be created using the following syntax.

Syntax:

```
app.filter('filtername', function()
{
   return function (input,optionalparam1,optionalparam2)
   {
      var output;
      //custom filter code goes here
      return output;
   }
})
```

Program 4.7: Program for custom filter.

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
```

```
</script>
</head>
<body>
   <div ng-app="app">
         <div ng-controller="userCtrl as user">
         <p>
         {{ user.username | makeUppercase }}
         </p>
         </div>
   </div>
   <script>
         var app = angular.module('app', []);
         app.filter('makeUppercase', function ()
     {
         return function (item)
     {
         return item.toUpperCase();
         };
         });
         app.controller('userCtrl', function ()
     {
         this.username = 'Gajanan Deshmukh';
         });
   </script>
</body>
</html>
```

Output:

GAJANAN DESHMUKH

- In above example, we have created filter makeUppercase which converts the item to uppercase and returns the value from it.

4.3 AngularJS FORMS

- AngularJS facilitates you to create a form enriches with data binding and validation of input controls.
- Input controls are ways for a user to enter data. A form is a collection of controls for the purpose of grouping related controls together.
- In angular, a form is a collection of controls. Controls (such as, input, select, textarea, button) are the ways to interact with the form.

- In AngularJS, we use ng-model to access
- property using ng-model.
- ng-model binds input fields of the f
- synchronously, means the value in the
- view will be same.

4.3.1 Working with AngularJS F

- AngularJS forms are commonly used t
- application provides this facility. An
- easily create the forms and apply them
- Following are the form controls used

1. **Input element:** For accepting inp
   for binding input fields. The nam
   to the screen with the binding ex

2.

```
</div>
<div ng-swit
```

In AngularJS, we use ng-model to access user's data. We bind input field to a model property using ng-model.

ng-model binds input fields of the form to the object of the model ($scope) synchronously, means the value in the scope object of the model and in the form of view will be same.

4.3.1 Working with AngularJS Forms

AngularJS forms are commonly used to accept information from user. Almost every application provides this facility. AngularJS provides us all the infrastructure to easily create the forms and apply them with proper event handling and validations.

Following are the form controls used in AngularJS forms:

1. **Input element:** For accepting input values, type will be text and ng-model is used for binding input fields. The name entered in the input field will be displayed on to the screen with the binding expression {{ }}.

Example:

```
<form>
   Enter the Name: <input type="text" ng-model="name">
</form>
<p>the value entered is {{name}}
```

2. **Select elements:** bind select boxes to your application with the ng-model directive. The property defined in the ng-model attribute will have the value of the selected option in the select box.

Example:

```
<form>
   Select a Fruit:
   <select ng-model="myVar1">
      <option value="Apple">Apple
      <option value="Banana">Banana
      <option value="Mango">Mango
      <option value="PineApple">PineApple
   </select>
   <div ng-switch="myVar1">
      <div ng-switch-when="Apple">
         <p>you selected apple</p>
      </div>
      <div ng-switch-when="Banana">
```

```
            <p>you selected banana</p>
        </div>
        <div ng-switch-when="Mango">
            <p>you selected mango</p>
        </div>
        <div ng-switch-when="PineApple">
            <p>you selected pineapple</p>
        </div>
</div>
</form>
```

3. **Radio button elements:** Bind radio buttons to our application with the ng-model directive. Radio buttons with the ng-model can have different values, but only the selected one will be used.

Example:

```
<form>
    Select the city
    <input type="radio" ng-model="myVar" value="Pune">Pune
    <input type="radio" ng-model="myVar" value="Mumbai">Mumbai
    <input type="radio" ng-model="myVar" value="Nashik">Nashik
    <input type="radio" ng-model="myVar" value="Solapur">Solapur
    <div ng-switch="myVar">
        <div ng-switch-when="Pune">
            <p>You selected pune city</p>
        </div>
        <div ng-switch-when="Mumbai">
            <p>You selected Mumbai city</p>
        </div>
        <div ng-switch-when="Nashik">
            <p>You selected Nashik city</p>
        </div>
       <div ng-switch-when="Solapur">
            <p>You selected Solapur city</p>
        </div>
    </div>
</form>
```

4. **TextArea elements:** It is used to define multiline text input field. It can hold unlimited number of characters. AngularJS modifies the behavior of TextArea element but we have to use ng-model with it.

Example:

```
<form>
    <textarea ng-model="Textarea1"></textarea>
    <p>The content of the textarea=</p>
    <h1>{{Textarea1}}</h1>
</form>
```

5. **Checkbox elements:** A checkbox has the value true or false. Apply the ng-model directive to a checkbox, and use its value in our application.

Example:

```
<form>
  <div>
      Gender <input type="checkbox" ng-model="myVar">
      <h1 ng-show="myVar">Checked</h1>
  </div>
</form>
```

Following example will illustrate student information form will contain all the form elements.

Program 4.8: Write a program to createAngularJS Form using form Controls.

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js
"></script>
<body>
<div ng-app="">
<form name="form">
<h1>
<center>Student Information form</center>
</h1>
Enter your name<input type="text" ng-model="size" name="size" ng-
required="true" />
```

```
<span ng-show="form.size.$error.required">The alue is required!</span>
<br>
Gender <input type="checkbox" ng-model="myVar">Male
<input type="checkbox" ng-model="myVar1">FeMale
<p ng-show="myVar">Male Checked</p>
<p ng-show="myVar1">Female Checked</p>
<br>
Select the state
<input type="radio" ng-model="myVar" value="Maharashtra">Maharashtra
<input type="radio" ng-model="myVar" value="Delhi">Delhi
<input type="radio" ng-model="myVar" value="Kolkata">Kolkata
<input type="radio" ng-model="myVar" value="Bengaluru">Bengaluru
<div ng-switch="myVar">
<div ng-switch-when="Maharashtra">
<p>You selected Maharashtra</p>
</div>
<div ng-switch-when="Delhi">
<p>You selected Delhi </p>
</div>
<div ng-switch-when="Kolkata">
<p>You selected Kolkata</p>
</div>
<div ng-switch-when="Bengaluru">
<p>You selected Bengaluru</p>
</div>
</div>
<br>
Select a city:
<select ng-model="myVar1">
<option value="Pune">Pune
<option value="Mumbai">Mumbai
<option value="Nagpur">Nagpur
<option value="Aurangabad">Aurangabad
</select>
<div ng-switch="myVar1">
<div ng-switch-when="Pune">
<p>you selected Pune city</p>
</div>
<div ng-switch-when="Mumbai">
```

```
<p>you selected Mumbai city</p>
</div>
<div ng-switch-when="Nagpur">
<p>you selected Nagpur city</p>
</div>
<div ng-switch-when="Aurangabad">
<p>you selected Aurangabad city</p>
</div>
</div>
<br>
Enter detailed Address<textarea ng-model="address">
</textarea>
<p>your address {{address}}</p>
</form>
</body>
</html>
```

Output:

Student Information form

Enter your name Prajakta

Gender ☐ Male ☐ FeMale

Male Checked

Female Checked

Select the state ◉ Maharashtra ○ Delhi ○ Kolkata ○ Bengaluru

You selected Maharashtra

Select a city: Pune

you selected Pune city

Enter detailed Address kalewadi

your address kalewadi

4.3.2 Model Binding

- Model binding is a very powerful feature of AngularJS. It allows us two-way data binding model. It is the synchronization between the model and view.
- When the model changes, the view reflect the change, also when the view is changes the model is updated. It happens immediately and automatically.
- ng-model is a directive in AngularJS that represents model and its primary purpose is to bind the view and the model.
- The ng-model attribute is used for,
  - Binding controls such as input, text area and selects in the view into the model.
  - Provide a validation behavior - for example, a validation can be added to a text box that only numeric characters can be entered into the text box.
  - The ng-model attribute maintains the state of the control (By state, we mean that the control and the data is bound to be always kept in sync. If the value of our data changes, it will automatically change the value in the control and vice versa)

Program 4.9: Program for model binding.

```
<!DOCTYPE html>
<html>
<head>
<title>TextArea Form</title>
</head>
<body>
<script src="https://code.angularjs.org/1.6.9/angular.js"></script>
<div ng-app="myApp" ng-controller="userCtrl">
<form>
      <textarea rows="4" cols="50" ng-model="desc"></textarea><br><br>
</form>
</div>
<script>
var app = angular.module('myApp',[]);
app.controller('userCtrl', function($scope){
$scope.desc="This is simple text area form control in angularJS"});
</script>
</body>
</html>
```

Output:

```
This is simple text area form control in angularJS
```

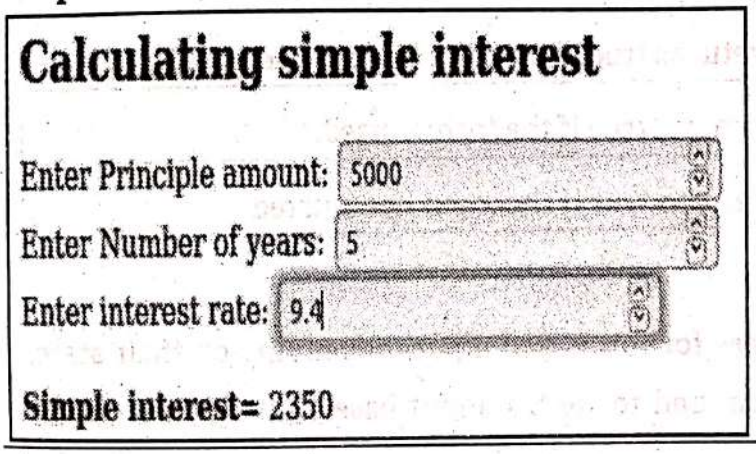
Explanation:

- The ng-model directive is used to attach the member variable called "desc" to the "textarea" control. The "desc" variable will actually contain the text, which will be passed on to the text area control.
- Here we are attaching the member variable to the scope object called "desc" and putting a string value to the variable.
- Let's try to have another example which will calculate simple interest and we will use three text fields which will bound with three ng model directives.

Program 4.10: Program for simple interest.

```
<!DOCTYPE html>
<html>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"><
/script>
<body>
<div ng-app="" ng-init="principle=5000;no_of_years=5;rate=9.4">
<h2>Calculating simple interest</h2>
Enter Principle amount: <input type="number" ng-model="principle"><br>
Enter Number of years: <input type="number" ng-model="no_of_years"><br>
Enter interest rate: <input type="number" ng-model="rate"><br>
<p><b>Simpleinterest=</b> {{principle*no_of_years*rate/100}}</p><br>
</div>
</body>
</html>
```

Output:



4.3.3 Form Controller

- Form controller keeps track of all its control and nested forms as well as the state of them, such as being valid/invalid or dirty/pristine. Each form directive creates an instance of form controller.
- Angular provides properties on forms that help us to validate them. They give us various information about form or its input applied to a form.
- Following are the methods of form controller:
  1. **$rollbackviewvalue():** It cancel an update and reset the input element's value to prevent an update to the `$modelValue`.
  2. **$commitviewvalue():** It commit a pending update to the $model value.
  3. **$getControls():** This method returns a shallow copy of the controls. The controls can be instances of formcontroller, and of NgModelController. If you need access to the controls of child-forms, we have to call $getControls() recursively on them.
  4. **$removecontrols():** It is used to deregister a control from the form.
  5. **$setDirty():** It is used to set the form to dirty state.
  6. **$setPristine():** It is used to set the form to pristine state.
  7. **$setUntouched():** Sets the form to its untouched state.
  8. **$setSubmitted():** Sets the form to its $submitted state. This will also set `$submitted` on all child and parent forms of the form.
- Following are the properties of form controller.

Table 4.1: Properties of Form Controller

Property	Class	Description
$valid	ng – valid	It returns true if item state is currently valid.
$invalid	ng – invalid	It returns true if the item state is invalid according to rules.
$pristine	ng – pristine	It returns true if the form has not been used yet.
$dirty	ng – dirty	It returns true if the form is used.
$touched	ng – touched	It returns true if the input has blurred.

4.3.4 Using CSS Classes

- AngularJS provides some CSS classes for forms and inputs depending on their state. It also allows us for styling of form and to control input based on the state of the form.

- **Following are the different CSS classes used by AngularJS:**
1. **ng-touched:** It used to check whether the field has been touched or not.
2. **ng-untouched:** It used to check whether field has not been touched.
3. **ng-pristine:** It used to find whether field has not been modified.
4. **ng-dirty:** It used to check whether field has been modified.
5. **ng-valid:** It used to checks whether field is valid.
6. **ng-invalid:** It used to checks whether field is invalid .
7. **ng-submitted:** AngularJS set this CSS class if the form is submitted.

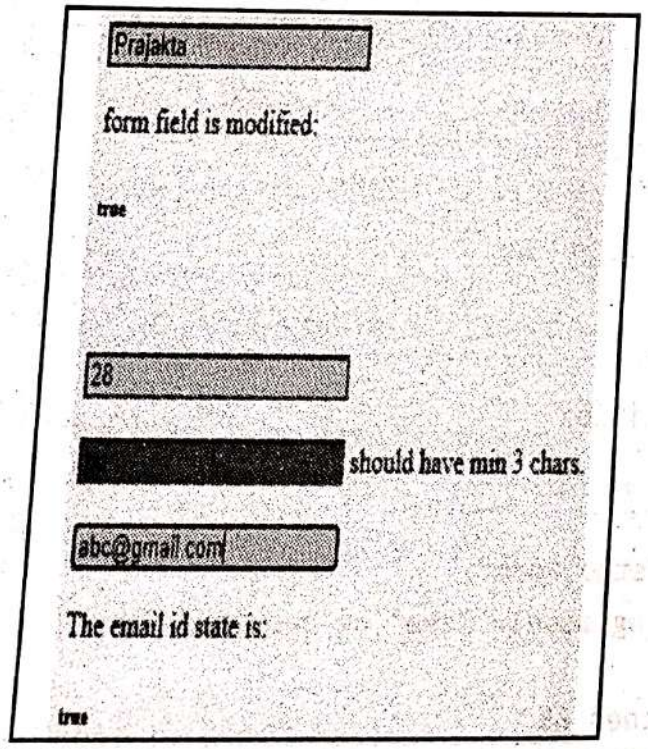
- We have to provide the proper implementation of these CSS classes in the program. AngularJS will automatically include them depending on the state.

Program 4.11: Program to demonstrate CSS classes.

```
<!DOCTYPE html>
<html lang="en">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<head>
<script src="https://code.angularjs.org/1.6.9/angular.js"></script>
<style>
   input.ng-pristine {
   background-color: pink;
   }
   input.ng-touched.ng-invalid {
   background-color: red;
   }
   input.ng-dirty {
   background-color: yellow
   };
   input.ng-touched.ng-valid {
   background-color: green;
   }
</style>
<title>Student Registration form</title>
</head>
<body ng-app="" >
   <form name="Form1" novalidate="stud form">
   <input type="text" name="Name" ng-model="Name" ng-required="true"
   placeholder="Enter Name" />
   <span ng-show="Form1.Name.$touched &&
   Form1.Name.$error.required">Name is required.</span><br/>
```

```
<p>form field is modified:</p>
<h6>{{Form1.Name.$dirty}}</h6><br><br>

<input type="number" name="age" ng-model="age" ng-required="true"
placeholder="Enter your age" />
<span ng-show="Form1.age.$touched && Form1.age.$error.required">age
is required.</span><br/><br/>

<input type="text" name="city" ng-minlength="3" ng-maxlength="15"
ng-model="city" ng-require="true" placeholder="Enter city">
<span ng-show="Form1.city.$touched &&
Form1.city.$error.minlength">should have min 3 chars.</span>
<span ng-show="Form1.city.$touched &&
Form1.city.$error.maxlength">should have Max 10 chars.</span><br>
<span ng-show="Form1.city.$touched &&
Form1.city.$error.required">city is required.</span><br />

<input type="email" id="email" ng-model="email" name="email"
placeholder="Enter Email id" ng-required="true" />
<span ng-show="Form1.email.$touched &&
Form1.email.$error.email">Please enter valid email></span><br>
<p>The email id state is:</p>
<h6>{{Form1.email.$valid}}</h6><br>
</form>
</body>
</html>
```

Output:



4.3.5 Form Events

Event handling plays very important role in every application programming. We can handle almost every type of event in AngularJS. Several ng events we will discuss here most common events are mouse, key, click etc.

1. **ng-click:** The ng-click directive allows you to specify custom behavior when an element is clicked.

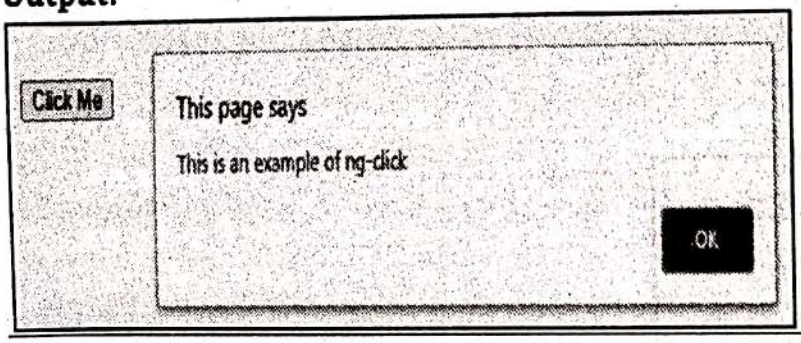
Syntax:

```
<element ng-click="expression">contents</element>
```

Program 4.12: Program for ng-click event

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.j
s"></script>
</head>
<body ng-app="clickExample">
<div ng-controller="userController">
<input type="button" ng-click="showAlert()" value="Click Me"></button>
</div>
<script>
var app = angular.module("clickExample", []);
app.controller('userController', ['$scope', function ($scope) {
$scope.showAlert = function () {
alert("This is an example of ng-click");
  }
  }]);
</script>
</body>
</html>
```

Output:



2. **ng-change event:** It is used to set a custom behavior for change event, the event is triggered when a form element's contents are changed.

Syntax:

```
<element ng-change="expression">contents</element>
```

Program 4.13: Program for ng change event

```
<!DOCTYPE html>
<html lang="en">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.j
s"></script>
</head>
<body ng-app="myApp">
<div ng-controller="userCtrl">
Are you developer <input type="checkbox" ng-model="isTrue" ng-
change="count=count+1" />
Count: {{count}}
<p>{{isTrue}}</p>
</div>
</div>
<script>
var app = angular.module("myApp", []);
app.controller('userCtrl', ['$scope', function ($scope) {
$scope.isTrue = true;
}]);
</script>
</body>
</html>
```

Output:

Are you developer ☑ Count: 8

true

9. **ng-copy and ng-paste:** These events are used to set custom behavior for copy and paste. We can find whether data has copied or pasted accordingly event will be triggered.

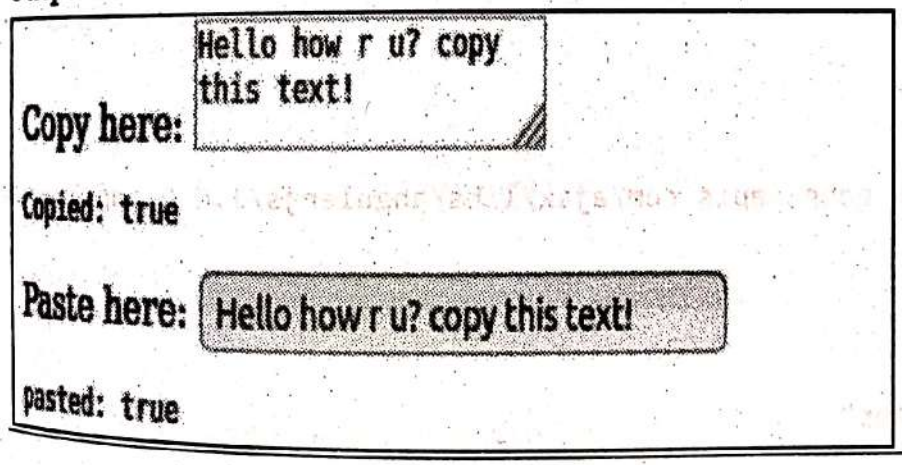
Syntax:

```
<element ng-copy="expression">contents </element>
<element ng-paste="expression">contents </element>
```

Program 4.14: Program for ng-copy and ng-paste event

```
<!DOCTYPE html>
<html>
<head>
<title></title>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.j
s"></script>
</head>
<body ng-app>
        <div ng-init="isCopy=false; copyValue='Hello how r u? copy this text!'">
        Copy here:<textarea ng-copy="isCopy=true" ng-
model="copyValue"></textarea>
        <br />
                <pre>Copied: {{isCopy}}</pre>
        </div>
        <div ng-init="isPaste=false">
                Paste here: <input ng-paste="isPaste=true" /><br />
                <pre>pasted: {{isPaste}}</pre>
        </div>
</body>
</html>
```

Output:



4. **ng-cut event:** This is also used as custom event. This element will be triggered when data is cut. We will see this event using example.

Program 4.15: Program for ng-cut event.

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.j
s"></script>
</head>
<body ng-app>
   <div ng-init="isCut=false; value='sample text !'">
   Cut the contents: <input ng-cut="isCut=true" ng-model="value" />
<br />
   <pre>Cut: {{isCut}}</pre>
   </div>
</body>
</html>
```

Output:

Cut the contents: sample text !

Cut: false

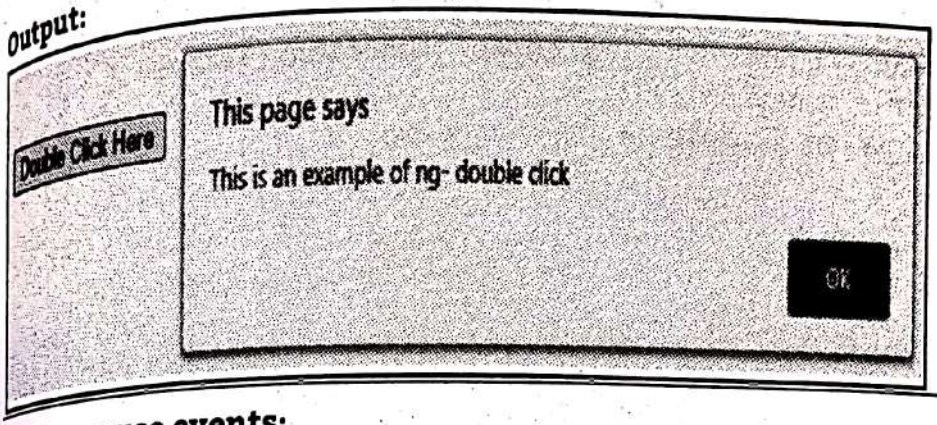
5. **ng-dblclick events:** To set a custom behaviour for dblclick event, the event is triggered when the element is double clicked using a mouse pointer.

Program 4.16: Program for ng-double click event

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.j
s">
</script>
</head>
<body ng-app="myApp">
```

```
<div ng-controller="userCtrl">
<input type="button" ng-dblclick="showAlert()" value="Double Click
Here"></button>
</div>
<script>
var app = angular.module("myApp", []);
app.controller('userCtrl', ['$scope', function ($scope) {
$scope.showAlert = function () {
alert("This is an example of ng- double click");
}
}]);
</script>
</body>
</html>
```

Output:



6. ng-mouse events:

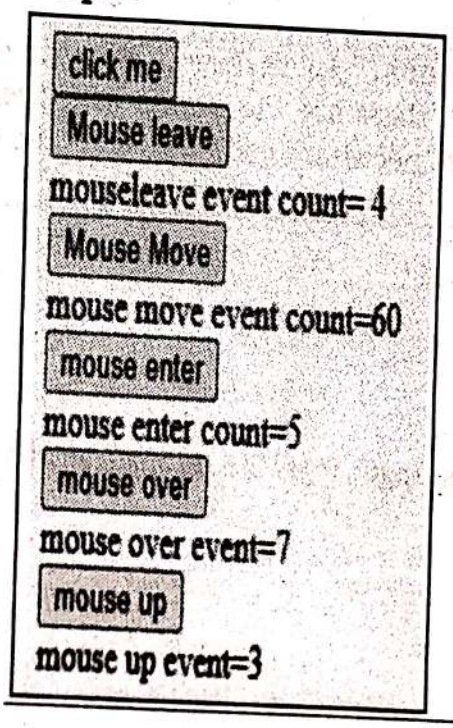
- These events are used to handle mouse events. Mouseleave, mousemove, mouseenter, mouseover, mouseup, mousedown these are the events which will trigger according to mouse events. we can write necessary code for each type of event.

Program 4.17: Program for ng-mouse event

```
<!DOCTYPE html>
<html>
<head>
  <script
  src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.j
  s">
  </script>
</head>
<body ng-app="myApp">
  <div ng-controller="userCtrl">
```

```
    <input type="button" ng-mousedown="alert()" value="click
me"></button><br>
    <input type="button" value="Mouse leave" ng-mouseleave="count1=count1+1"
/>
    <br>
    mouseleave event count= {{count1}}<br>
    <input type="button" value="Mouse Move" ng-mousemove="count2=count2+1"
/>
    <br>
    mouse move event count={{count2}}<br>
    <input type="button" value="mouse enter" ng-
mouseenter="count3=count3+1"/>
    <br>
    mouse enter count={{count3}}<br>
    <input type="button" value="mouse over" ng-mouseover="count4=count4+1"/>
    <br>
    mouse over event={{count4}}<br>
    <input type="button" value="mouse up" ng-mouseup="count5=count5+1"/><br>
    mouse up event={{count5}}<br>
    </div>
    <script>
      var app = angular.module("myApp", []);
      app.controller('userCtrl', ['$scope', function ($scope) {
      $scope.alert = function () {
      alert("This is an Example of ng-mousedown.");
      }
      }]);
   </script>
</body>
</html>
```

Output:



7. ng- keyup, ng-keydown events:

- The ng-keypress directive in AngluarJS is used to apply custom behavior on a keypress, keydown, keyup event. It is supported by <input>, <select> and <textarea> element.

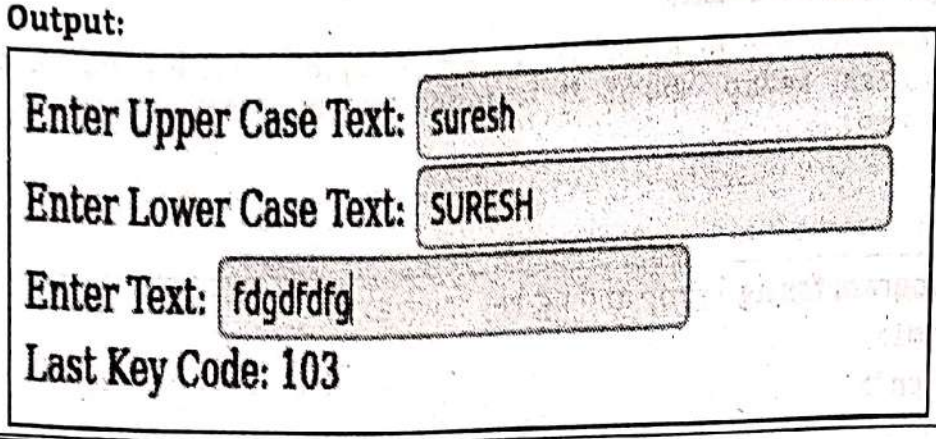
Syntax:

```
<element ng-keypress="expression">contents </element>
```

Program 4.18: Program for ng-keyup and ng-keydown event

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>
   <script
   src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.m
   in.js">
   </script>
</head>
<body ng-app="myApp">
   <div ng-controller="userCtrl">
      Enter Upper Case Text: <input type="text" ng-model="text" ng-
      keydown="lowerCase(text)" /><br />
      Enter Lower Case Text: <input type="text" ng-model="text1" ng-
      keyup="upperCase(text1)" /><br />
      Enter Text: <input type="text" ng-keypress="getkeys($event)" nd-
      model="ftext"><br />
                                          Last Key Code: {{keyval}}
   </div>
   <script>
      var app = angular.module("myApp", []);
      app.controller('userCtrl', ['$scope', function ($scope) {
      $scope.lowerCase = function (text) {
      $scope.text = text.toLowerCase();
      $scope.upperCase = function (text1) {
      $scope.text1 = text1.toUpperCase();
      $scope.getkeys = function (event) {
      $scope.keyval = event.keyCode;
      }
      }
      }]);
   </script>
</body>
</html>
```

Output:



4.3.6 Custom Model Update Triggers

- By default, change in content of the input element triggers the model update and validates the form. However, we can override this behavior by attaching ng-model-options directive with event name to the input element.
- We can use these update triggers with the options where update and debounce where changes will be updated after specific time.
- Mouse up update will be performed when mouse will be clicked on input field and clicked similarly mouse down update will be performed when mouse will be clicked down on field.
- It can be implemented in following way:

```
ng-model - options = "{ updateOn : 'blur'}"
```

Following are the values of ng-model-options:

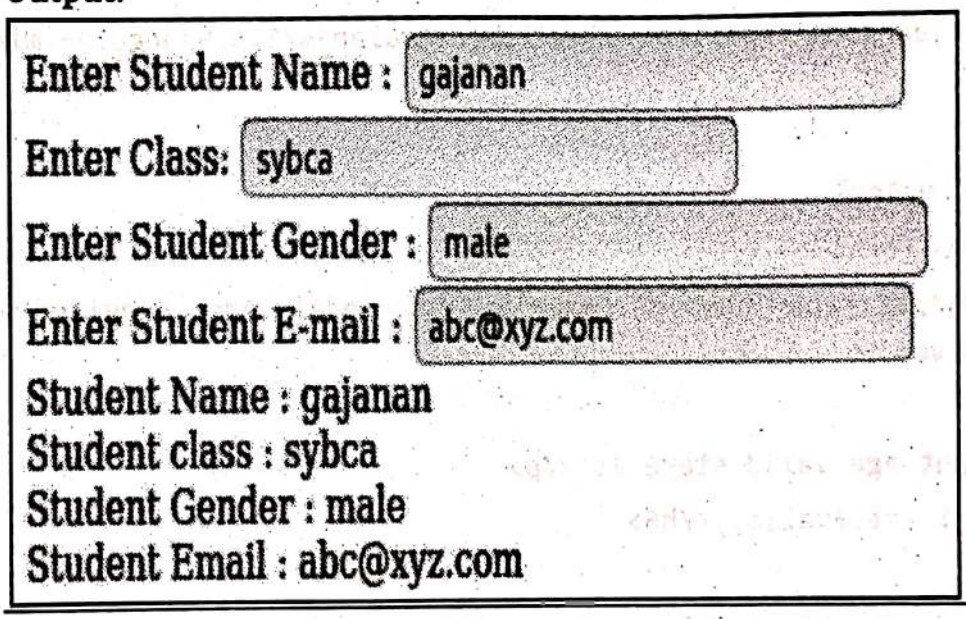
1. **{updateOn : 'blur'}:** Updates the model when this element loose focus.
2. **{updateOn : 'mouseup'}:** It updates the model when this element is clicked and mouse button is released.
3. **{updateOn : 'default'}:** It updates the model with default behavior, as and when the key is pressed.
4. **{updateOn : 'default mousedown'}:** It updates the model with default behavior, as and when key is pressed or mousedown event is fired.
5. **{debounce limit}:** It updates the model after the specific time limit.

Program 4.19: Program to demonstrate custom model update triggers.

```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.j
s">
</script>
</head>
```

```
<body ng-app="myApp">
   <div ng-controller="userCtrl">
   <formnovalidate>
      Enter Student Name : <input type="text" ng-model="student.name"
      ng-model-options="{updateOn:'blur'}" /></br>
      Enter Class: <input type="text" ng-model="student.class" ng-
      model-options="{updateOn : 'mouseup'}" /><br>
    Enter Student Gender : <input type="text" ng-model="student.gender"
                              ng-model-options="{debounce:2000}" /></br>
    Enter Student E-mail : <input type="email" ng-model="student.email"
        ng-model-options="{updateOn:'blur',debounce:{blur:0} }" /></br>
   </form>
   <span>Student Name : {{student.name}}</span><br>
   <span>Student class : {{student.class}}</span><br>
   <span>Student Gender : {{student.gender}}</span><br>
   <span>Student Email : {{student.email}}</span><br>
   <script>
   var app = angular.module('myApp', []);
   app.controller('userCtrl', function ($scope) {
   $scope.student = {};
   });
   </script>
</body>
</html>
```

Output:



- In the above example, the class name will be displayed when we will click on the input field with mouse up event. Similarly email will be updated with debounce update after two seconds.

4.3.7 Custom Validation

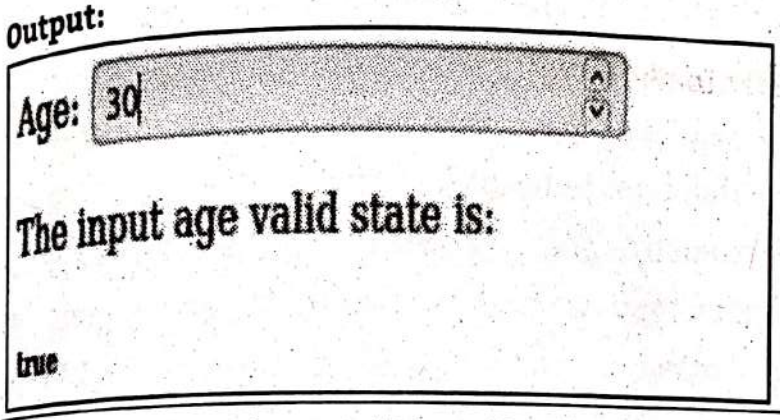
- To create your own validation function add a new directive to your application, and deal with the validation inside a function with certain specified arguments.
- In the appDirective, we have made the ng-model directive mandatory using "require" keyword as we are going to perform validation based on model value of the input element.
- Then we are using "link" keyword that is used to call a function for the directive. This link function defines APIs and functionality is necessary for the custom directive. This link function is executed for each instance of the directive.
- This link function accepts four parameters by default and they are:
  - **scope:** The scope of the directive.
  - **element:** The element that this directive attached to.
  - **attributes:** A hash object with key-value pairs of normalized attribute (first character lower case and each consecutive word name's first character in upper case) names.
  - **control:** The control (object) it is working with.

Program 4.20: Program for custom validation.

```
<!DOCTYPE html>
<html lang="en">
<head>
<script src=
    "https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.j
s">
</script>
<body ng-app="myApp">
    <form name="form1">
        Age: <input type="number" name="age" ng-model="age" required app-
        directive>
    </form>
    <p>The input age valid state is:</p>
    <h6>{{form1.age.$valid}}</h6>
    <script>
        var app = angular.module('myApp', []);
        app.directive('appDirective', function() {
        return {
```

```
require: 'ngModel',
link: function(scope, element, attr, userCtrl) {
function myValidation(value) {
if (value >=18) {
userCtrl.$setValidity('charE', true);
}
else {
userCtrl.$setValidity('charE', false);
}
return value;
}
userCtrl.$parsers.push(myValidation);
}
};
});
</script>
</body>
</html>
```

Output:



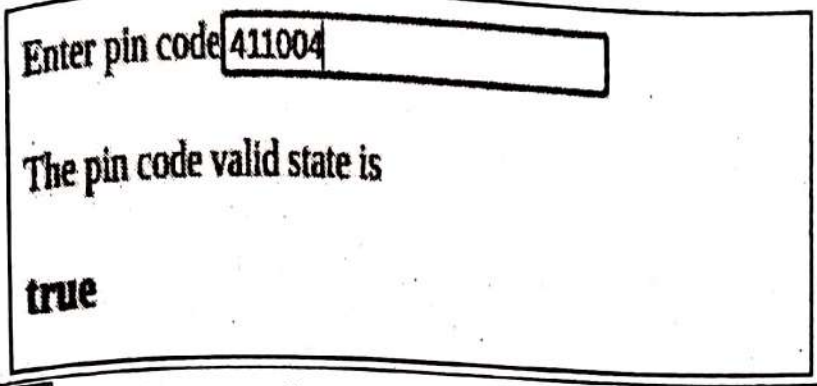
- For better understanding we have another example to accept pin code from user if the length is greater than 5 then state will be valid.

Program 4.21: Program for Pin Code validation.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.j
s">
```

```
</script>
<script>
    angular.module('myApp', []);
    angular.module('myApp', []).directive('customValidation', function
() {
    return {
    restrict: 'A',
    require: 'ngModel',
    link: function (scope, element, attr, ctrl) {
    function validationError(value)
    {
    if (value.length > 5)
    {
    ctrl.$setValidity('validstate', true);
    }
    else {
    ctrl.$setValidity('validstate', false)
    }
    return value;
    }
    ctrl.$parsers.push(validationError);
    }
    };
    });
</script>
</head>
<body ng-app="myApp">
    <div>
    <form name="myform">
    Enter pin code<input name='test' type='text' required ng-
    model='test' custom-validation></td>
    <br><br> The pin code valid state is
    <h3>{{myform.test.$valid}}</h3>
    </form>
    </div>
</body>
</html>
```

Output:



4.3.8 $http Services

- The $http service is a core AngularJS service that facilitates communication with the remote HTTP servers via the browser's XMLHttpRequest object or via JSON.
- The AngularJS provides a control service named as AJAX – $http, which serves the task for reading all the data that is available on the remote servers.
- The demand for the requirement of desired records gets met when the server makes the database call by using the browser. The data is mostly needed in JSON format. This is primarily because for transporting the data, JSON is an amazon method and also it is straightforward & effortless to use within AngularJS, JavaScript, etc.
- There are various methods that can be used to call $http service, this are also shortcut methods to call $http service.

1. **$http.get():** This method is used to get data from requested url.
2. **$http.post():** This method is used to send data to requested url.
3. **$http.head():** This method is used to get data from requested url with headers.
4. **$http.put():** This method is used to send data to specified url.
5. **$http.delete():** This method is used to delete resource from specified url.
6. **$http.jsonp():** This method is used to get data from specified url in json format.
7. **$http.patch():** This method is used to perform patch request from specified.

Program 4.22: Program to create a service to calculate area and circumference of circle.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script
    src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.j
    s"></script>
<script>
    var myApp = angular.module('myApp', []);
```

```
myApp.service('Circle', function () {
var pi = 3.14;
this.Area = function (r) {
return pi*r*r;
}
this.Circuimference = function (r) {
return 2*pi*r;
}
});
myApp.controller("userCtrl", function ($scope, Circle) {
$scope.AreaofCircle = Circle.Area(5);
$scope.CircuimferenceOfCircle = Circle.Circuimference(5);
});
</script>
</head>
<body ng-app="myApp">
<div ng-controller="userCtrl">
The area of circle is {{AreaofCircle}}<br>
The Circumference of circle is {{CircuimferenceOfCircle}}<br>
</div>
</body>
</html>
```

Output:

The area of circle is 78.5
The Circumference of circle is 31.400000000000002

4.4 AJAX IMPLEMENTATION USING $HTTP

- AJAX is the short form of Asynchronous JavaScript and XML. Ajax was primarily designed for updating parts of a web page, without reloading the whole page.
- The reason for designing this technology was to reduce the number of round trips which were made between the client and the server and the number of entire page refresh which used to take place whenever a user required certain information.
- AJAX allowed web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This meant that it was possible to update parts of a web page, without reloading the whole page.

```
n () {
) {
ion ($scope, Circle) {
5);
cle.Cir
```

was primarily
hole page.
number of round trips
the number of entire page
equired certain information.
ously by exchanging small amounts
eant that it was possible to update
age.

- Many modern day web applications actually follow this technique for refreshing the page or getting data from the server.
- Angular provides two different APIs to handle Ajax requests which are :
  1. **$resource:** "$resource" property in Angular, which is used to interact with servers at a high level.
  2. **$http:** The $http is another AngularJS service which is used to read data from remote servers. The most popular form of data which is read from servers is data in the JSON format.

Program 4.23: Program for Ajax implementation of $http service.

app.js

```
var myApp = angular.module('myApp', []);
myApp.controller('myController', function($scope, $http){
$http.get('sample_json.js')
.then(function (response){
$scope.jsondata = response.data;
console.log("status:" + response.status);
}).catch(function(response) {
console.error('Error occurred:', response.status, response.data);
}).finally(function() {
console.log("Task Completed.");
});
});
```

Sample.json

```
[
{
"RollNo" : "1",
"name" : "Suresh",
"class": "FyBca",
"City" :"pune"
},
```

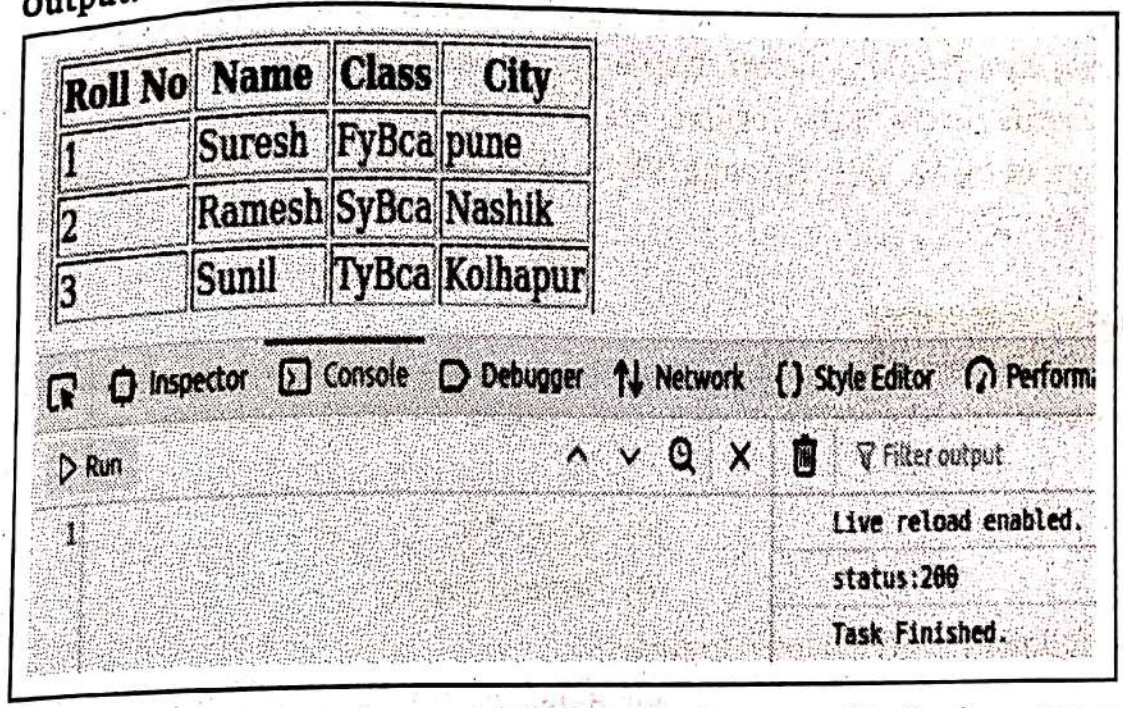
```
{
"RollNo" : "2",
"name" : "Ramesh",
"class": "SyBca",
"City" :"Nashik"
},
{
"RollNo" : "3",
"name" : "Sunil",
"class": "TyBca",
"City" :"Kolhapur"
}
]
```

httpexample.html

```
<html>
<head>
<script src=
"https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
/script>
<script src="app.js"></script>
</head>
<body>
<div ng-app="myApp" ng-controller="myController">
<table border="1">
<tr>
      <th>Roll No</th>
      <th>Name</th>
      <th>Class</th>
      <th>City</th>
```

```
    </tr>
    <tr ng-repeat="data in jsondata">
        <td>{{ data.RollNo}}</td>
        <td> {{data.name}} </td>
        <td>{{data.class}}</td>
        <td>{{data.City}}</td>
    </tr>
    </table>
    </div>
    </body>
    </html>
```

Output:



- In app.js file using console.log() method we are displaying status of the form and message task finished.

Summary

- AngularJS provides us several filters for the formatting of data.
- Uppercase filter converts the data to uppercase characters similarly lowercase filter converts uppercase characters into lowercase.
- Currency filter allows us to display numbers in $, rupees format.

- Date filter allows us to display date in format like mm/dd/yy, or short, medium, long date.
- Order by filter requests and numbers in the numerical request. Sometimes built-in filters in AngularJS can not meet the needs for filtering the output. In such situation custom filter can be created which can pass the output in the required manner.
- AngularJS provides the client-side validation for input fields as well as forms. It also provides the information about whether the input field is modified or not etc. Some common properties of input field are $dirty, $invalid, $valid, $pristine, $pending, $submitted, $error.
- AngularJS gives you very common validation tokens which are email, max, min, maxlength, minlength, number, pattern, required etc.
- There are several form events which we have covered like ng-click, ng-dbl-click, ng-change, mouse events as well as key events.
- $http service makes a request to the server, and returns a response. There are several methods for $http service like get(), put(), post(), patch(), delete(), head(), patch().
- AngularJS framework provides the $http service and is used to do the HTTP request to retrieve the desired data from the server. The server would then make a database call to retrieve the records in the JSON format. We can send AJAX requests in various ways through angular AJAX calls through $http service, JSON calls through $http service and REST API calls.

Check Your Understanding

1. Which directive binds the value of application data to HTML input controls in AngularJS?
   (a) ng-app (b) ng-model
   (c) ng-bind (d) None of above
2. Which of the following in a filter in AngularJS?
   (a) currency (b) date
   (c) uppercase (d) All of above
3. How to combine filter with an expression?
   (a) `{{expression,pipe}}` (b) `{{expression.pipe}}`
   (c) `{{expression| pipe}}` (d) `{{expression / pipe}}`
4. Which directive we use to specify angular application ?
   (a) ng –app (b) ng- application
   (c) ng –controller (d) None of above

5. Which directive we use to specify angular Controller ?
   (a) ng -Controller (b) ng- control
   (c) angular – controller (d) ng – Ctrl
6. Which service is used to connect with remote server ?
   (a) `$service` (b) `$http`
   (c) `$route` (d) `$interval`
7. Which function is used to create a module ?
   (a) `app.module()` (b) `angular.module()`
   (c) `angular.create()` (d) `app.createModule()`
8. Which of the following are validation directives ?
   (a) ng –required (b) ng – minlength
   (c) ng – pattern (d) All of Above
9. Which directive we use to specify angular application ?
   (a) ng –app (b) ng- application
   (c) ng –controller (d) None of above
10. Which of the following directive allows us to use forms?
    (a) ng –include (b) ng -form
    (c) ng –bind (d) ng -attach
11. Which of the following is not a valid angular filter?
    (a) lowercase (b) orderby
    (c) email (d) currency
12. Which angular directive is used to bind the values of HTML controls (input, select, textarea) to application data ?
    (a) ng –Interval (b) ng -bind
    (c) ng –model (d) ng-timeout
13. AngularJS directives are used in .....
    (a) service (b) module
    (c) component (d) view
14. Which of the following is valid AngularJS expression?
    (a) {{ 4 +5 }} (b) { 4+5}
    (c) (4+5) (d) ({ 4+5})
15. What is the purpose of $http?
    (a) It is an inbuilt service used to make a ajax call to get the server data.
    (b) It is the http protocol
    (c) It generates hypertext language
    (d) None of above

ANSWERS

1. (b)	2. (d)	3. (c)	4. (a)	5. (a)	6. (b)	7. (b)	8. (d)	9. (a)	10. (b)
11. (c)	12. (c)	13. (d)	14. (a)	15. (a)					

Practice Questions

Q. I Answer the following questions in short.

1. Explain date filter with example?
2. Define services in AngularJS?
3. How can someone make an AJAX call using AngularJS?
4. Explain lowercase filter?
5. Explain uppercase filter?
6. Explain filter in filter?
7. Explain order by filter?
8. List validations in AngularJS?

Q. II Answer the following questions.

1. Explain different types of filters in AngularJS?
2. What are AngularJS forms ? Explain its elements?
3. List various AngularJS form validations?
4. What are different types of form events?
5. What are custom model update triggers? Justify?
6. What is custom validation?
7. What is $http service?
8. Create a hello world application program using AngularJS.
9. Write an AngularJS program to create service for finding factorial and display it.

Q. III Define the following terms.

1. Filter
2. Service
3. $http Service
4. Event handling

***

5... Dependency Injection, Services

Objectives...

- To understand what is dependency injection.
- To cover how to use AngularJS services.
- To understand how to use built-in service.
- To understand and create custom service in AngularJS.

5.1 INTRODUCTION

- With AngularJS we are able to create services; it provides us to isolate business logic of every component of application. Also, we can use the framework's dependency injection mechanism to easily supply any component with a desired dependency. This framework comes with built in services which are very useful in development purpose.
- AngularJS is powered with dependency injection mechanism that manages life cycle of each component. This mechanism is responsible for creating and distributing components within the application.
- In this chapter, we will discuss how dependency injection is used, how we can create services, also how built-in services are used in AngularJS.

5.2 WHAT IS DEPENDENCY INJECTION?

- Dependency Injection is one of the best features of AngularJS. It is a software design pattern in which objects are passed as dependencies. It helps us to remove hard coded dependencies and makes dependencies configurable. Using Dependency Injection, we can make components maintainable, reusable and testable.

- Dependency Injection is required for the following:
  - Separating the process of creation and consumption of dependencies.
  - It allows us to create independent development of the dependencies.
  - We can change the dependencies when required.
  - It allows injecting mock object as dependencies for testing.
- Entire service concept in AngularJS is heavily dependent on and driven by dependency injection system. Any service known to AngularJS can be simply injected into any other service, directive, or controller by stating it as a dependency.
- AngularJS will automatically figure out what the service is, what it further depends on, and create the entire chain before injecting a fully instantiated service.
- Dependency Injection is a concept that started more on the server side, to basically propagate reuse, modularity, and testability of code.
- Dependency Injection states that instead of creating an instance of a dependent service when we need it, the class or function should ask for it instead. Something else (usually known as an injector) would then be responsible for figuring out how to create it and pass it in.
- Dependency Injection allows us to:
  - Change the underlying implementation of a dependency without manually changing each dependent function.
  - Change the underlying implementation just for the test, to prevent it from making server calls.
  - Explicitly state what needs to be included and present before this function or constructor can execute.

Program 5.1: Program to demonstrate how dependency works.

```
<!DOCTYPE html >
<html ng-app="myapp">
<head>
<title>Angular Services</title>
<script
src="http://ajax.googleapis.com/ajax/libs/AngularJS/1.6.9/angular.min.js"><
/script>
<script>
   var module = angular.module('myapp', []);
   module.controller("MyController", function ($scope, $window) {
   $scope.winWidth = $window.innerWidth;
   });
</script>
</head>
<body ng-controller="MyController">
   <p>Window width: {{winWidth}}px</p>
</body>
</html>
```

Output:

Window width: 1366px

- Here we haven't instantiated this window service ourselves. The angular dependency management subsystem takes care of this behind scene. This technique is called **dependency injection.**

5.3 UNDERSTANDING SERVICES

- AngularJS services are a set of tightly related functions, managed by angular framework, which are made readily available for use across an application.
- Service in AngularJS is a function or an object that can be used to share data and the behavior across the application (controller, directives, filters, other services etc.) or we can say services in AngularJS are objects that are wired together using dependency injection and they can be used to share and organize code across the application.
- Services are defined using service() functions and they can be injected into other AngularJS object like controller, directives, filters, other services, etc.
- AngularJS services exist to provide functionality that's available to the entire application. A service in AngularJS is any piece of common functionality that can be shared across entire application.
- In AngularJS, service is a JavaScript object which contains a set of functions to perform certain tasks. Services are created by using service() function on a module and then injected into controllers.
- Service provides us methods to keep data across the lifetime of the angular app.
- It provides us method to communicate data across the controllers in a consistent way.
- Services are singleton objects that gets instantiated only once per application (by the $injector) and lazy loaded (created only when necessary).
- It is used to organize and share data and functions across the applications.

5.4 USING BUILT-IN SERVICE

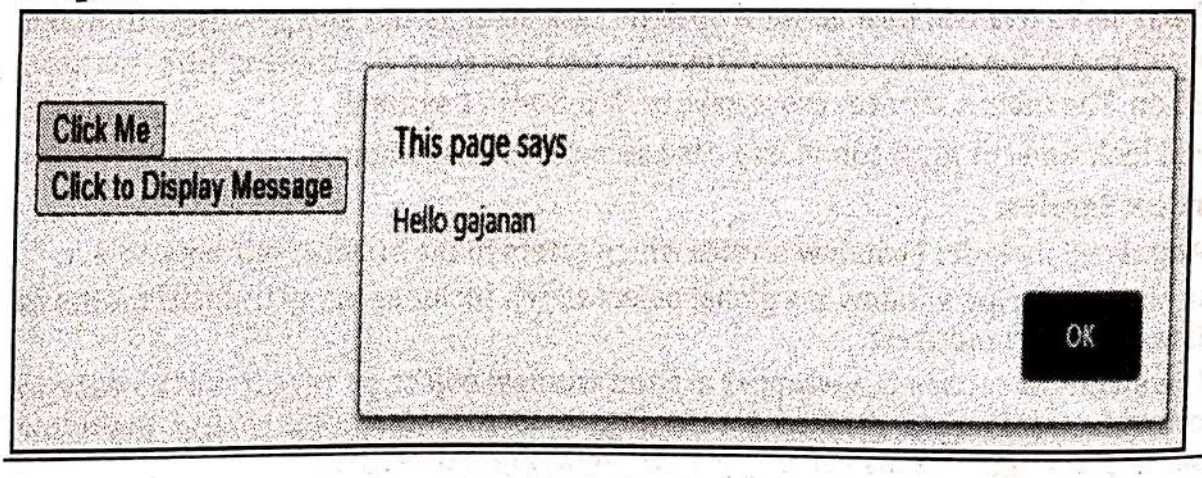
- AngularJS provides several built-in services which can be used easily in the program. In runtime, these services are automatically registered with the dependency injector, so you can easily incorporate them into your angular applications by using dependency injection. AngularJS provides many built-in services.
- There are about 30 built-in services, for example, $http, $route, $window, $location, etc. Each service is responsible for a specific task such as the $http is used to make AJAX call to get the server data, the $route is used to define the routing information, and so on. The inbuilt services are always prefixed with $ symbol.
- We will look some of most commonly used services in this chapter.

1. **$Window Service:**
   - AngularJS includes $window service which refers to the browser window object.
   - In the JavaScript, window is a global object which includes many built-in methods like alert(), prompt() etc.
   - The $window service is a wrapper around window object, so that it will be easy to override, remove or mocked for testing. It is recommended to use $window service in AngularJS instead of global window object directly.

Program 5.2: Program for $windows service.

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/AngularJS/1.6.9/angular.min.js">
</script>
</head>
<body ng-app="myApp" ng-controller="userCtrl">
<button ng-click="ButtonClick('Hello World!')">Click Me</button><br>
<button ng-click="DisplayMessage()">Click to Display Message</button>
<script>
    var myApp = angular.module('myApp', []);
    myApp.controller("userCtrl", function ($scope, $window) {
    $scope.ButtonClick = function (message) {
    $window.alert(message);
    }
    $scope.DisplayMessage = function () {
    var name = $window.prompt('Enter Your Name');
    $window.alert('Hello ' + name);
    }
    });
  </script>
</body>
</html>
```

Output:



2. $location service:

- The `$location` service has methods which return information about the location of the current web page.

Program 5.3: Program for $location service.

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/AngularJS/1.6.9/angular.min.js">
</script>
<body>
<div ng-app="myApp" ng-controller="userCtrl">
<p>The url of this page is:</p>
<h3>{{text1}}</h3>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('userCtrl', function($scope, $location) {
$scope.text1 = $location.absUrl();
});
</script>
</body>
</html>
```

Output:

The url of this page is:

file:///C:/Users/waghm/Desktop/a.html#!/

3. $interval Service:

- AngularJS includes $interval service which performs the same task as setInterval() method in JavaScript. The $interval is a wrapper for setInterval() method, so that it will be easy to override, remove or mocked for testing.
- The $interval service executes the specified function on every specified milliseconds duration.

Program 5.4: Program for $interval service.

```
<!DOCTYPE html>
<html >
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/AngularJS/1.6.9/angular.min.js">
</script>
</head>
<body ng-app="myApp">
```

```
<div ng-controller="userCtrl">Counter will increment after two seconds:
{{counter}}</div>
<script>
    var myApp = angular.module('myApp', []);
    myApp.controller("userCtrl", function ($scope, $interval) {
    $scope.counter = 0;
    var increaseCounter = function () {
    $scope.counter = $scope.counter + 1;
    }
    $interval(increaseCounter, 2000);
    });
</script>
</body>
</html>
```

Output:

Counter will increment after two seconds: 3

4. $timeout service:

- This $timeout service of AngularJS allows the developer to set some time delay before execution of the function.
- To use the $timeout service we must first get it injected into a controller function.

Program 5.5: Program for $timeout service.

```
<!DOCTYPE html>
<html>
<head>
<script src=
"http://ajax.googleapis.com/ajax/libs/AngularJS/1.6.9/angular.min.js">
</script>
<script type="text/javascript">
    var app = angular.module('myApp', []);
    app.controller('userCtrl', function($scope, $timeout) {
    $scope.text = "Welcome Greetings of the day."
    $timeout(function() {
    $scope.Message=
    "this message hasbeen displayed after 4 secons on $timeout call
    function";
    }, 4000);
    });
</script>
</head>
```

```
<body>
<div ng-app="myApp" ng-controller="userCtrl">
<p>Timeout Message will be displayed here:</p>
<b>{{Message}}</b>
</div>
</body>
</html>
```

Output:

Timeout Message will be displayed here:

this message hasbeen displayed after 4 secons on $timeout call function

5. $filter Service:

- $filter service is used for formatting the data displayed to the user. The $filter service works with AngularJS built-in filter like uppercase, lowercase, currency, orderby etc. It can be used with custom filter also.

Syntax:

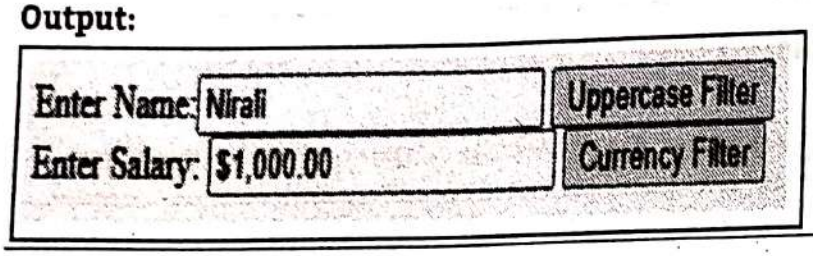
```
$filter('uppercase') ($scope.name);
$filter ('currency') ($scope.salary);
```

Program 5.6: Program for $filter service

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://ajax.googleapis.com/ajax/libs/AngularJS/1.6.9/angular.min.js">
</script>
<meta charset="utf-8">
<script>
   var app = angular.module('MyApp', []);
   app.controller("FilterController", function ($scope, $filter) {
   $scope.name = '';
   $scope.salary = 0;
   $scope.UppercaseFilter = function () {
   $scope.name = $filter('uppercase')($scope.name);
   }
   $scope.CurrencyFilter = function () {
   $scope.salary = $filter('currency')($scope.salary);
   }
   });
</script>
</head>
<body>
```

```
<div ng-app="MyApp" ng-controller="FilterController">
Enter Name:<input type="text" ng-model="name">
<button ng-click="UppercaseFilter()">Uppercase Filter</button></br>
Enter Salary: <input type="text" ng-model="salary">
<button ng-click="CurrencyFilter()">Currency Filter</button></br>
</div>
</body>
</html>
```

Output:



6. $document service:

- It is a wrapper for the browsers window object. It is a jQuery collection. When we use $document into AngularJS application it returns jQuery collection that contains document object and its properties.

Program 5.7: program for $document service.

```
<!DOCTYPE html>
<html>
<head>
<title>angular document example</title>
<script
src="http://ajax.googleapis.com/ajax/libs/AngularJS/1.6.9/angular.min.js">
</script>
<meta charset="utf-8">
<script>
angular.module('MyApp',[]).controller('ExampleController',['$scope',
'$document', function ($scope, $document) {
$scope.title = $document[0].title;
$scope.windowTitle = angular.element(window.document)[0].title;
}]);
</script>
</head>
<body ng-app="MyApp" ng-controller="ExampleController">
<p>$document title: <b ng-bind="title"></b></p>
<p>window.document title: <b ng-bind="windowTitle"></b></p>
</body>
</html>
```

Output:

$document title: angular document example

window.document title: angular document example

7. $log service:

- This service is used to write errors, warnings and for debugging onto the browser console.

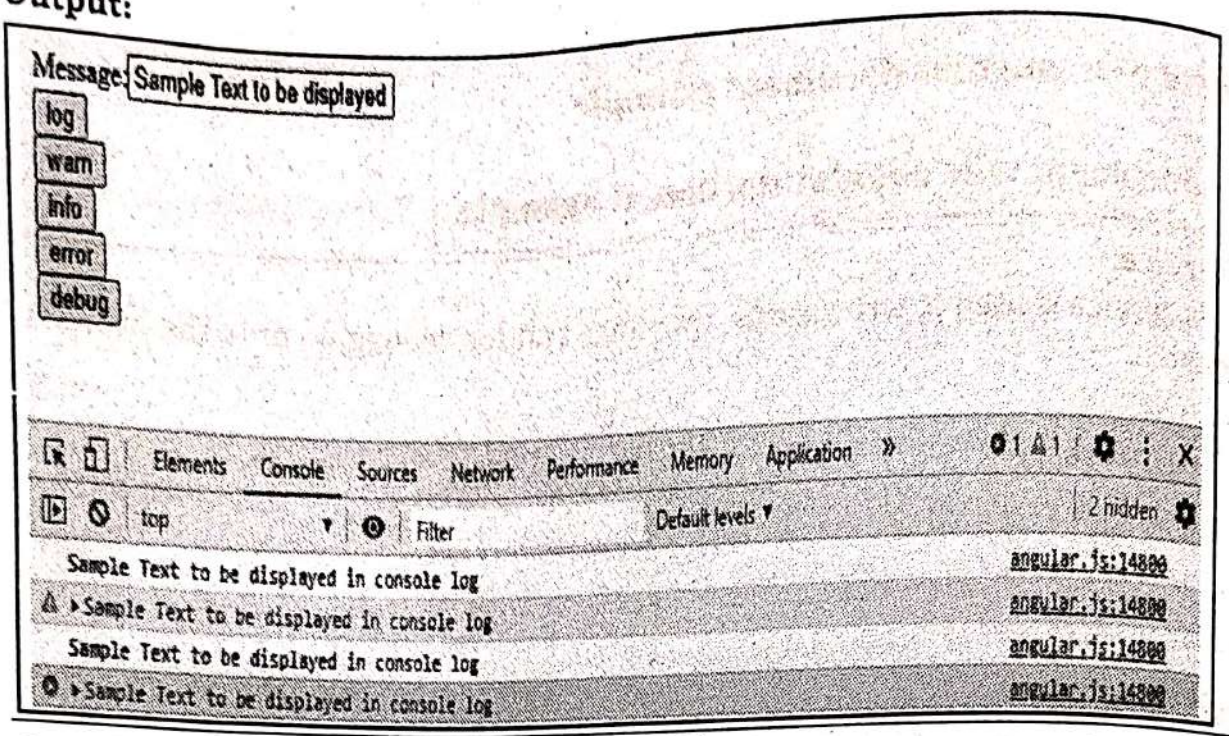
Syntax:

```
$log.log('message');
$log.error('Message');
$log.info('Message');
$log.warn('Message');
$log.debug('Message');
```

Program 5.8: Program for $log service.

```
<!DOCTYPE html>
<html>
<script
src="//ajax.googleapis.com/ajax/libs/AngularJS/1.6.9/angular.min.js">
</script>
<meta charset="utf-8">
<script>
angular.module('MyApp', [])
.controller('userCtrl', ['$scope', '$log', function($scope, $log) {
$scope.$log = $log;
$scope.message = 'Sample Text to be displayed in console log';
}]);
</script>
</head>
<body>
<div ng-app="MyApp" ng-controller="userCtrl"><br>
<label>Message:<input type="text" ng-model="message" /></label><br>
<button ng-click="$log.log(message)">log</button><br>
<button ng-click="$log.warn(message)">warn</button><br>
<button ng-click="$log.info(message)">info</button><br>
<button ng-click="$log.error(message)">error</button><br>
<button ng-click="$log.debug(message)">debug</button><br>
</div>
</body>
</html>
```

Output:



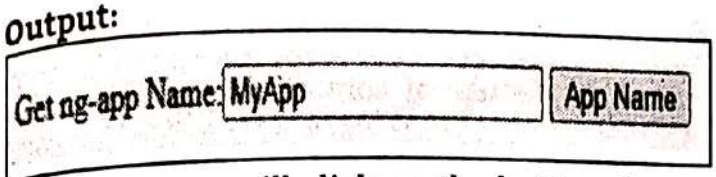
8. $root service:

- This is the element where ng-app was declared. The element represents the root element of application. It is also the location where the application's $injector service gets published, and can be retrieved using `$rootElement.injector()`.

Program 5.9: Program for $root service.

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://ajax.googleapis.com/ajax/libs/AngularJS/1.6.9/angular.min.js">
</script>
<script>
var app = angular.module('MyApp', []);
app.controller("userCtrl", function ($scope, $rootElement) {
$scope.myfunction = function () {
$scope.name = $rootElement.attr('ng-app')
}
});
</script>
</head>
<body ng-app="MyApp" ng-controller="userCtrl">
Get ng-app Name:<input type="text" ng-model="name">
<button ng-click="myfunction()">App Name</button></br>
</body>
</html>
```

Output:



- When user will click on the button the app module name will be displayed in input field.

- Apart from this there are other built in services which are as follows:
  - **$anchorScroll:** It provides the capability to scroll a page anchor specified in `$location.hash()` based on the rules defines in HTML5 specifications.
  - **$animate:** It provides animation hooks to link into both CSS and JavaScript based animations.
  - **$cacheFactory:** It provides the capability to put key/value pairs into an object cache where they can be retrieved later by other code components using the same service.
  - **$compile:** It provides the capability to compile an HTML string or a DOM object into a template and produce a template function that can link the scope and template together.
  - **$cookies:** It provides read and write access to the browsers cookies.
  - **$document:** It specifies a JQuery wrapped reference to the browsers window document element.
  - **$exceptionHandler:** It specifies a handler to which uncaught exceptions in AngularJS expressions are delegated.
  - **$interpolate:** It compiles a string with markup into an interpolation function.
  - **$locale:** It provides localization rules that are consumed by various AngularJS Components.
  - **$parse:** It parses an AngularJS expression string to a JavaScript function.
  - **$q:** It provides a promise/deferred implementation service.
  - **$resource:** It enables us to create an object that can interact with RESTful server-side data source.
  - **$root element:** It provides access to the root element in the AngularJS Application.
  - **$rootScope:** It provides access to the root scope for the AngularJS application.
  - **$route:** It provides deep – linking URL support for controllers and views by watching the $location.url() and mapping the path to existing route definitions.
  - **$routeParams:** It provides a service that enables us to retrieve the current set of parameters in the route.
  - **$sanitize:** It provides a service that can be used to sanitize input by parsing the HTML into tokens.
  - **$sce:** It provides strict contextual escaping functionality when handling data from untrusted sources.
  - **$swipe:** It provides a service that makes implementing device swipe types of directives easier.
  - **$templateCache:** It provides the capability to read templates from a web server into a cache for later use.

5.5 CREATING CUSTOM SERVICE

- We saw how to use AngularJS services through the use of some built-in AngularJS services.
- We should consider creating an AngularJS service, if what we are implementing falls into one of the broad criteria:
  - **It needs to be reusable:** More than one controller or service will need to access the particular function that is being implemented.
  - **Application level state:** Controllers get created and destroyed. If we need state stored across our application, it belongs in a service.
  - **It is independent of the view:** If what we are implementing is not directly linked to a view, it probably belongs in a service.
  - **It is independent of the view:** If what we are implementing is not directly linked to a view, it probably belongs in a service.
  - **It integrates with a third-party service:** We need to integrate a third-party service. And it is achieved by services.
- In Angular JS we can create our own service by using module object like below:

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

- Here myApp is reference of the module object. This module object provides two different functions to create user defined service.

```
service();
factory();
```

- we can create a custom service using service() function like below:

```
myApp.service("servicenane", functionName){
// we have to declare variables using this keyword
}
```

- To use this service in controller, we need to give service details when we will be creating controller.

```
MyApp.controller("controllerName", function($scope, serviceName){
//here you can access members of service.
}}
```

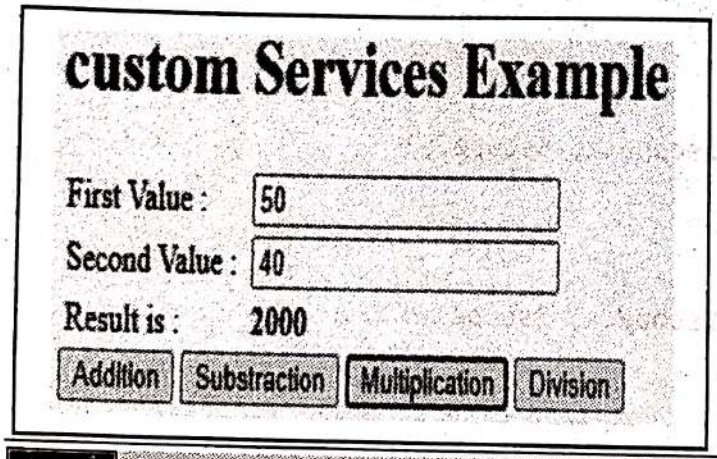
Program 5.10: Program for AngularJS Custom Service:

```
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/AngularJS/1.6.9/angular.min.j
s"></script>
```

```
<script>
var myApp = angular.module("myApp", []);
myApp.service("mathService", function() {
this.add = function (x, y) {
return parseInt(x) + parseInt(y);
}
this.sub = function (x, y) {
return parseInt(x) - parseInt(y);
}
this.mul = function (x, y) {
return parseInt(x) * parseInt(y);
}
this.div = function (x, y) {
return parseInt(x) / parseInt(y);
}
})
myApp.controller("mathController", function ($scope, mathService) {
$scope.x = 50;
$scope.y = 40;
$scope.result = 0;
$scope.calcAdd = function () {
$scope.result = mathService.add($scope.x, $scope.y)
}
$scope.calcSub = function () {
$scope.result = mathService.sub($scope.x, $scope.y)
}
$scope.calcMul = function () {
$scope.result = mathService.mul($scope.x, $scope.y)
}
$scope.calcDiv = function () {
$scope.result = mathService.div($scope.x, $scope.y)
}
})
</script>
```

```
<body ng-app="myApp">
<h1>custom Services Example</h1>
<div ng-controller="mathController">
<table>
<tr><td>First Value :<td>
<td><input type="text" ng-model="x" /><td></tr>
<tr><td>Second Value :<td><td><input type="text" ng-model="y" />
<td></tr>
<tr><td>Result is :<td>
<td><b>{{result}}</b></td>
<td></tr>
</table>
<input type="button" ng-click="calcAdd()" value="Addition" />
<input type="button" ng-click="calcSub()" value="Substraction" />
<input type="button" ng-click="calcMul()" value="Multiplication" />
<input type="button" ng-click="calcDiv()" value="Division" />
<br>
</div>
</body>
</html>
```

Output:



5.6 INJECTING DEPENDENCY IN SERVICE

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

- Dependency injection mainly reduces the tight coupling of code and creates modular code that is more maintainable and testable. AngularJS services are the objects that can be injected in any other Angular construct (like controller, filter, directive etc). You can define a service which does certain tasks and inject it wherever you want. In that way you are sure your tested service code works without any glitch.
- Following are the core components which can be injected into each other as dependencies.

1. constant:

- A constant can be injected everywhere. A constant can not be intercepted by a decorator, it means that the value of a constant should never be changed.

Syntax: `module.constant(name,constant);`

Program 5.11: Program to demonstrate the use of constant.

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://ajax.googleapis.com/ajax/libs/AngularJS/1.6.9/angular.min.js">
</script>
<script>
var myApp = angular.module('myApp', []);
myApp.constant('appSettings', {
APP_NAME: "Constant Demo App"
});
myApp.controller('constantController', ['$scope', 'appSettings', function
($scope, appSettings) {
$scope.appName = appSettings.APP_NAME;
}]);
</script>
</head>
<body class="container" ng-app="myApp" ng-controller="constantController">
<h1>Custom Service Through constant() Example</h1>
{{appName}}
</body>
</html>
```

Output:

Custom Service Through constant() Example

Constant Demo App

2. Value:

- A value is nothing more than a simple injectable value. The value can be a string or number. Value differs from constant in that value can not be injected into configurations, but it can be intercepted by decorators.
- It provides primitive value, function or object that may contain functions.

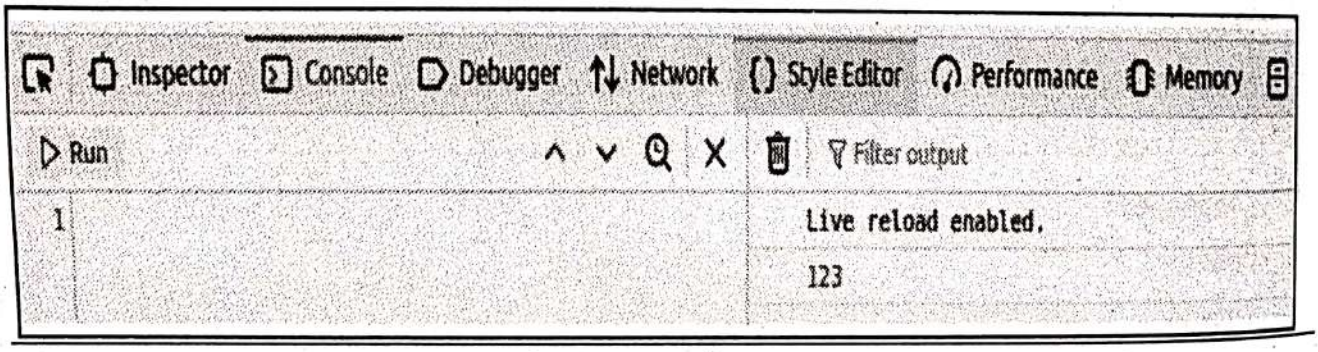
Syntax:

```
Module.value(name,value);
```

Program 5.12: Program to demonstrate the use of value in services.

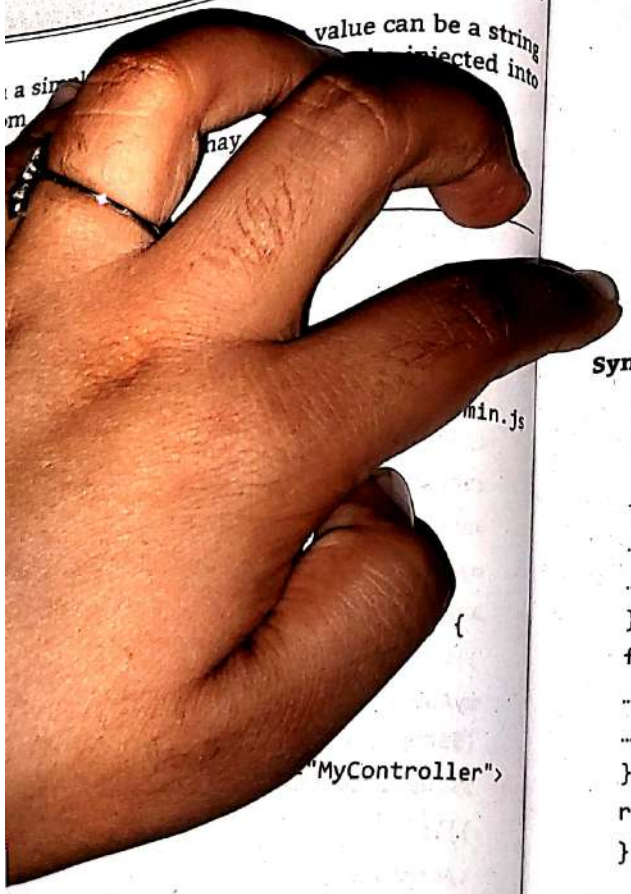
```
<!DOCTYPE html>
<html>
<head>
<script
src="http://ajax.googleapis.com/ajax/libs/AngularJS/1.6.9/angular.min.js
">
</script>
<script>
var myModule = angular.module("myModule", []);
myModule.value("numberValue",123);
myModule.controller("MyController", function($scope, numberValue) {
console.log(numberValue);
}); </script>
</head>
<body class="container" ng-app="myModule" ng-controller="MyController">
</body>
</html>
```

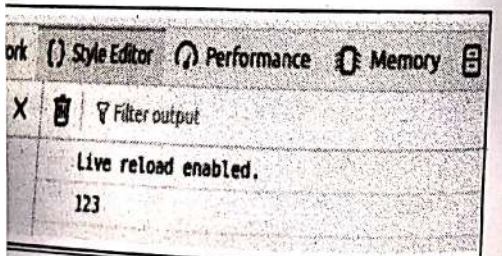
Output:



hrough constant() Example

value can be a string
jected into





3. **Factory:**
   - Factory is a function that creates values. When a service, controller etc. needs a value injected from a factory, the factory creates the value on demand. Once created, the value is reused for all services, controllers etc. which need it injected.
   - Thus, a factory differs from a value in that it can use a factory function to create the object it returns. You can also inject values into a factory for use when creating the object. You cannot do that with a value. It provides a function that returns a value, typically an object with function or single function.
   - A factory is an injectable function. A factory is a lot like a service in the sense that it is a singleton and dependencies can be specified in the function.
   - The difference between a factory and a service is that a factory injects a plain function so AngularJS will call the function and a service injects a constructor. A constructor creates a new object so new is called on a service and with a factory you can let the function return anything you want.
   - When you create a service using module.factory(), return value of function is passed as second parameter becomes the service object that AngularJS registers and injects later to other service/controller.

Syntax:

```
module.factory('Demoservice', function(){
var factory ={};

factory.firstMethod = function(){
.....
.....
}
factory.secondMethod = function(){
.....
.....
}
return factory;
});
```

```
Module.factory(name,function(dependencies) {….});
```

Program 5.13: Program to demonstrate the use of factory.

```
<!DOCTYPE html>
<html>
<head>
```

```
<script
src="http://ajax.googleapis.com/ajax/libs/AngularJS/1.6.9/angular.min.js
">
</script>
<script>
var myApp = angular.module('myApp', []);
myApp.controller('ServiceController', ['$scope', 'myService', function
($scope, myService) {
$scope.obj = myService.obj;
}]);
myApp.factory('myService', function () {
return {
obj: {
name: 'Gajanan Deshmukh',
Gender:'Male',
}
}
});
</script>
</head >
<body ng-app="myApp" ng-controller="ServiceController">
<h4>Custom Service Through factory() Example</h4>
Name is:{{ obj.name }}<br />
Gender is:{{ obj.Gender }}
</body>
</html >
```

Output:

Custom Service Through factory() Example

Name is:Gajanan Deshmukh
Gender is:Male

4. **Provider:**
   - Provider function that supports service configuration and defines $get method that returns configured object with function properties of single function.

     `Module.provider(name,providerfunc());`
   - Let's see an example for provider custom service.

Program 5.14: Program to implement the provider function.

```
<!DOCTYPE html>
<head>
<script
src="http://ajax.googleapis.com/ajax/libs/AngularJS/1.6.9/angular.min.js">
</script>
<script>
var myApp=angular.module('myApp',[]);
myApp.config(function($provide){
$provide.provider('user',function(){
this.$get=function(){
return{
name:"Gajanan"
}
};
});
}).controller('providerController',['$scope','user',function($scope,user){
$scope.name=user.name;
}]);
</script>
</head>
<body ng-app="myApp" ng-controller="providerController">
<h4>Custom AngularJS service through Provider()</h4>
Hello {{name}}
</body>
</html>
```

Output:

Custom angularJS service through Provider()

Hello Gajanan

5. **Service:**

Using `module.service('servicename',function(){})`

- When you create a service using module.service(), an instance of the function() passed as second parameter becomes the service object that AngularJS registers and injects later to other services/ controller.
- It has following syntax:

```
module.service('DemoService', function(){
this.firstMethod= function (){
.....
.....
}
this.secondMethod = function(){
.....
.....
}
});
```

- Let's look at a simple example of how you can create your own service. We are going to create a simple square service which makes square of two numbers.

Program 5.15: Program to implement Dependency Injection through service.

```
<! DOCTYPE html>
<head>
<meta charset="UTF-8">
<title>Service Example</title>
</head>
<script
src="https://ajax.googleapis.com/ajax/libs/AngularJS/1.6.9/angular.min.js">
</script>
<body>
<div ng-app="myApp" ng-controller="userCtrl">
<p>Result: {{result}}</p>
</div>
<script>
var myApp = angular.module("myApp", []);
myApp.service('SquareService', function () {
this.Power = function (a, b) {
```

```
return a * b;
}
});
myApp.controller('userCtrl', function ($scope, SquareService) {
$scope.result = SquareService.Power(3, 4);
});
</script>
</body>
</html>
```

Output:

Result: 12

5.6.1 Difference between Factory, Service and Provider

- **Factory:** When you're using a Factory you create an object, add properties to it, then return that same object. When you pass this service into your controller, those properties on the object will now be available in that controller through your factory.
- **Service:** When you're using Service, it's instantiated with the 'new' keyword. Because of that, you'll add properties to 'this' and the service will return 'this'. When you pass the service into your controller, those properties on 'this' will now be available on that controller through your service.
- **Providers:** Providers are the only service you can pass into your .config() function. Use a provider when you want to provide a module-wide configuration for your service object before making it available.

Summary

- DI or Dependency Injection is a software design pattern that deals with how code gets hold of its dependencies.
- Service in AngularJS is a function or an object that can be used to share data and the behavior across the application.
- In AngularJS, service is a JavaScript object which contains a set of functions to perform certain tasks. Services are created by using service() function on a module and then injected into controllers.

- In AngularJS there are several built in services like $window $root, $log $interval.
- In AngularJS we can create our own service and connect it to module. These are referred as custom services.
- Angular JS provides high level of dependency injection mechanism. In five different ways using the Factory, Provider, Service, Constant and Value.
- Value – It is a simple java script object, which is required to pass values to the controller during configuration phase
- Factory – It is function which is used to return a value.
- Service – It is singleton java script object containing a set of functions to perform certain tasks. It Is defined using service() function and injected into controller.
- Provider – it is used by AngularJS Internally to create services, factory etc. during the configuration phase.
- Constant – it can be injected every where and its value can not get change.

Check Your Understanding

1. Which of the following is an in built service in AngularJS?
   (a) $http (b) $route
   (c) $windows (d) All of the above
2. Services are ..... objects.
   (a) stateless (b) stateful
   (c) Both (a) and (b) (d) None of the above
3. Which service is used to communicate with remove server?
   (a) $interval (b) $http
   (c) $route (d) $timeout
4. Which of the following is true about AngularJS services?
   (a) Services are JavaScript function.
   (b) Services are responsible to do specific tasks only.
   (c) Built-in services are always prefixed with $ symbol.
   (d) All of the above
5. What is the purpose of $http ?
   (a) It is an inbuilt service used to make AJAX call to get the server data.
   (b) It is the http protocol.
   (c) It is used to pass on data to another program.
   (d) None of above

6. Can we create our own services through AngularJS services?
   (a) No
   (b) Yes
7. What is the use of $root services?
   (a) It is used to display name of controller
   (b) It gives us name of module
   (c) It is used to access root elements
   (d) None of above
8. Which are following components through which dependency can be injected?
   (a) Service
   (b) Provider
   (c) Factory
   (d) All of above
9. What is the use of $location built in service?
   (a) It is used to display absolute location of file.
   (b) It is used to locate module.
   (c) It is used to locate controllers.
   (d) None of above
10. Check whether code is correct?

```
app.factory('TestFactory', function myTestFactory($rootScope, $http,
$location) {
return function myTestReusable() {
// processing goes here
};
});
```

(a) Yes (b) No

ANSWERS

1. (d)	2. (a)	3. (b)	4. (d)	5. (a)	6. (b)	7. (b)	8. (d)	9. (a)	10. (a)

Practice Questions

Q. I Answer the following questions in short.

1. List various inbuilt services in AngularJS?
2. What is dependency injection? Explain with example.
3. What is service in AngularJS?
4. How custom services are used in AngularJS?
5. What is AngularJS Factory?

Q. II Answer the following questions.

1. Explain services in AngularJS?
2. Explain different built-in angular services?
3. What is dependency Injection?
4. How to create custom services?
5. Distinguish between Factory, Service and Provider.
6. Write SPA (Single Page Application) in AngularJS to create e-learning system.
7. Using AngularJS create online ecommerce site.
8. Using AngularJS create a To Do list Application.

Q. III Define the following terms.

1. Dependency Injection
2. Services
3. Custom services.
4. Provider.
5. Factory.

***