

UNIT 1: AngularJS Core Concepts

Features of angular JS...



The core features of AngularJS are as follows –

- **Data-binding** – It is the automatic synchronization of data between model and view components.
- **Scope** – These are objects that refer to the model. They act as a glue between controller and view.
- **Controller** – These are JavaScript functions bound to a particular scope.
- **Services** – AngularJS comes with several built-in services such as \$http to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app.
- **Filters** – These select a subset of items from an array and returns a new array.
- **Directives** – Directives are markers on DOM elements such as elements, attributes, css, and more. These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives such as ngBind, ngModel, etc.



- **Templates** – These are the rendered view with information from the controller and model. These can be a single file (such as index.html) or multiple views in one page using *partials*.
- **Routing** – It is concept of switching views.
- **Model View Whatever** – MVW is a design pattern for dividing an application into different parts called Model, View, and Controller, each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel). The Angular JS team refers it humorously as Model View Whatever.
- **Deep Linking** – Deep linking allows 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.
- AngularJS is an open source full featured JavaScript framework developed by Google. It was designed to build dynamic web apps in the first place.
... AngularJS allows to bind data and inject eliminate the most part of the code in order to prevent writing it.
- Published by Google in 2009 AngularJS has become one of the most popular JavaScript frameworks till no
- AngularJS is a JavaScript framework written in JavaScript.
- AngularJS is distributed as a JavaScript file, and can be added to a web page with a script tag:
- `<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>`
- AngularJS is an open source full featured JavaScript framework developed by Google. It was designed to build dynamic web apps in the first place.
... AngularJS allows to bind data and inject eliminate the most part of the code in order to prevent writing it.

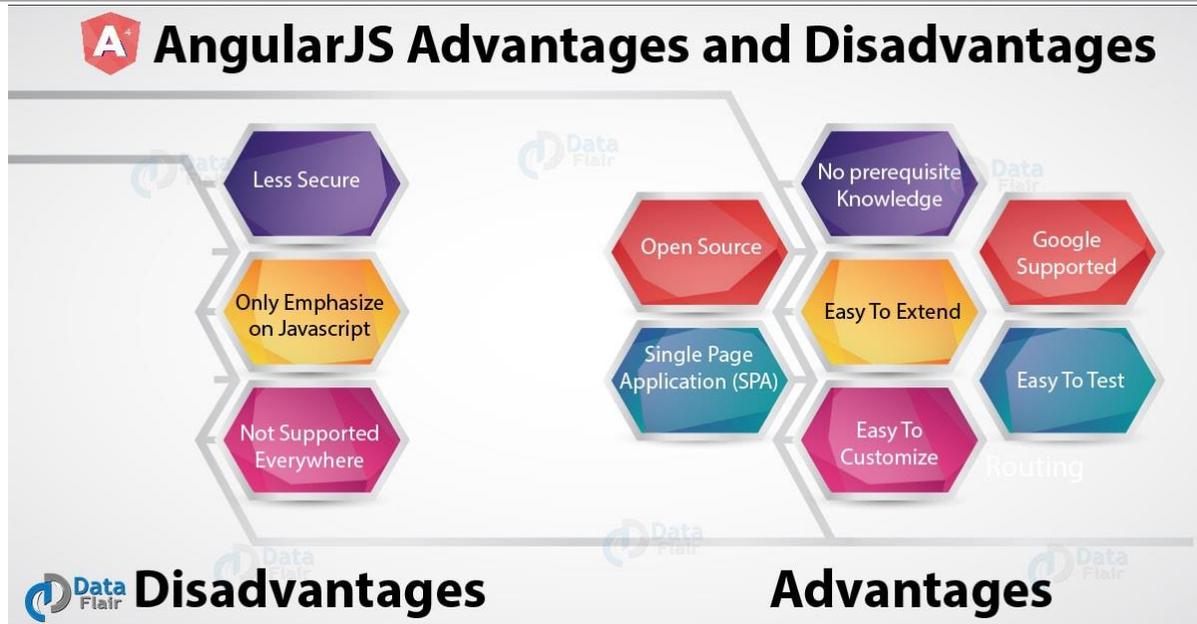
- AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly .



JAVASCRIPT VERSUS ANGULARJS

JavaScript is an interpreted programming language used to create dynamic web applications.	AngularJS is a front-end application framework based on JavaScript which excels at building dynamic and large single-page web apps (SPAs).
It's a client-side scripting language developed by Netscape Communications to power web apps.	It's an open-source JavaScript tool developed and maintained by Google with angular directives.
It's a full-featured language used to manipulate Document Object Model (DOM).	It extends HTML with new attributes along with other web technologies.
JavaScript expressions do not support filters.	AngularJS expressions support filters.
It validates the user input at the browser level before submitting the page to the server end.	It makes an ideal tool for any server technology.
It's the language of the web used to make websites more interactive.	It's a MVC based architecture that makes web apps easy to create from the scratch.

Difference Between .net



Advantages of AngularJS

The advantages of AngularJS are –

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

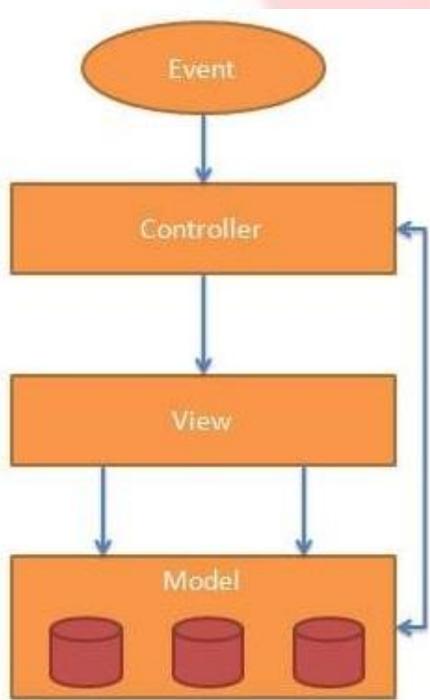
Disadvantages of AngularJS

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

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

AngularJS MVC Architecture

MVC stands for **Model View Controller**. It is a software design pattern for developing web applications. It is very popular because it isolates the application logic from the user interface layer and supports separation of concerns.



The MVC pattern is made up of the following three parts:

1. **Model:** It is responsible for managing application data. It responds to the requests from view and to the instructions from controller to update itself.



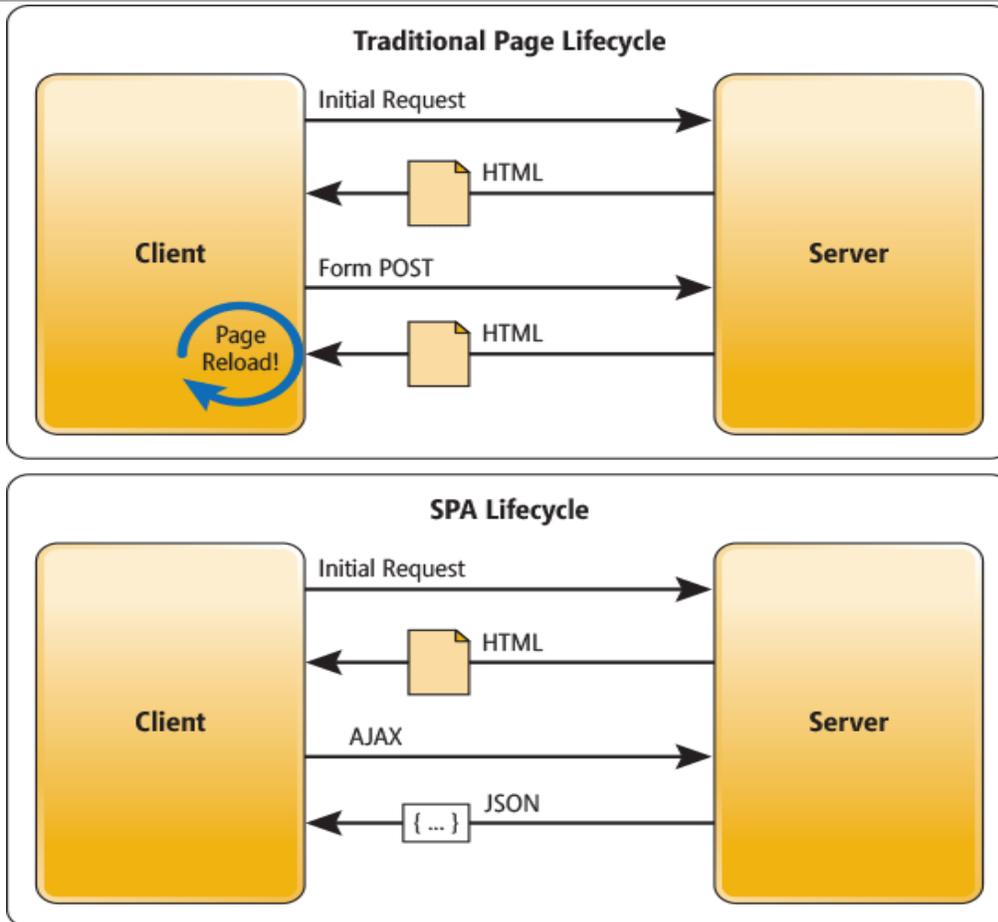
- View:** It is responsible for displaying all data or only a portion of data to the users. It also specifies the data in a particular format triggered by the controller's decision to present the data. They are script-based template systems such as JSP, ASP, PHP and very easy to integrate with AJAX technology.
- Controller:** It is responsible to control the relation between models and views. It responds to user input and performs interactions on the data model objects. The controller receives input, validates it, and then performs business operations that modify the state of the data model.

SPA:-

AngularJS is a powerful javascript framework, used for developing SinglePageApplication(SPA) projects in very clean and maintainable way.

(Single Page Application is a web application or website that interacts with the user by dynamically rewriting the current page rather than loading entire new pages from a server).

AngularJS is a powerful javascript framework, used for developing SinglePageApplication(SPA) projects in very clean and maintainable way.



(Single Page Application is a web application or website that interacts with the user by dynamically rewriting the current page rather than loading entire new pages from a server). Consider an example of GMAIL, where without refreshing a page, it opens all details of the page. It has only one HTML page, which requests for another page content from the server.

Angular extends Html DOM with new attributes like ng-model,ng-if, ng-bind, etc.

(Single Page Application is a web application or website that interacts with the user by dynamically rewriting the current page rather than loading entire new pages from a server). Consider an example of GMAIL, where without refreshing a page, it opens all details of the page. It has only one HTML page, which requests for another page content from the server.

Angular extends Html DOM with new attributes like ng-model,ng-if, ng-bind, etc.



Single Page Application

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. And navigation between pages performed without refreshing the whole page.

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. And navigation between pages performed without refreshing the whole page. AngularJs is a powerful javascript framework for building dynamic web applications. It became insanely popular nowadays. The good thing about Angular is that it has a set of ready-to-use modules to simplify building of single page applications.

SPA (Single Page Application). This means that the navigation within Netflix is performed without refreshing the whole page.

Angularjs is a Single Page Applications Framework. 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. And navigation between pages performed without refreshing the whole page.

Single-Page Applications (SPAs) are Web apps that load a single HTML page and dynamically update that page as the user interacts with the app. SPAs use AJAX and HTML5 to create a fluid and responsive Web apps, without constant page reloads. However, this means much of the work happens on the client side, in JavaScript.

Advantages of Single Page Applications

Given below are few advantages of Single Page Applications.

- Improved user experience.
- Web pages refresh faster as less bandwidth is being used.
- The deployment of the application – index.html, CSS bundle, and javascript bundle – in production becomes easier.
- Code splitting can be done to split the bundles into multiple parts.

Why Develop an SPA Using AngularJS?

Nowadays, there are many javascript applications that are available in the market like ember.js, backbone.js, etc. But still, a lot of web applications incorporate AngularJS in their development to create SPAs.

Given below are few reasons as for why AngularJS is a clear winner:

#1) No Dependencies

Unlike AngularJS, backbone.js has dependencies on underscore.js and jQuery. Whereas, ember JS has dependencies on handlebars and jQuery.

#2) Routing

Navigation between web pages built using AngularJS is very simple when compared to those that are built using other javascript frameworks. The directives used in AngularJS are lightweight, hence the performance metrics of AngularJS is appreciable.

#3) Testing

Once the application is built using AngularJS, automated testing could be performed for quality assurance using selenium. This is one of the awesome features of applications built using AngularJS development.



#4) Data Binding

AngularJS supports two-way data binding, i.e., whenever the model is updated, the view also gets updated as AngularJS follows MVC architecture.

Hence, the data can be viewed by the user, based on his preferences.

#5) Support for the Browser



AngularJS is supported in most of the browsers including IE version 9 and above. It can adapt to work on mobiles, tablets, and laptops too.

#6) Agility

AngularJS supports agility which means that it can cater to new requests from businesses as and when they come up into competitive work environments. The controllers can be implemented in the MVC architecture to service these requests.

There are around 30000 modules in AngularJS, that are readily available for rapid application development. When a developer wants to customize an existing application, he can use the modules that are already available and tweak the code instead of building the whole application from scratch.

Moreover, the contributors and experts in AngularJS are many in number, hence you would get quick responses to any queries that you post on discussion forums

In angular js some Important tag know to us (quick review of this)

1)div tag :- Definition and Usage. The <div> tag defines a division or a section or block in an HTML document. The <div> tag is used as a container for HTML elements

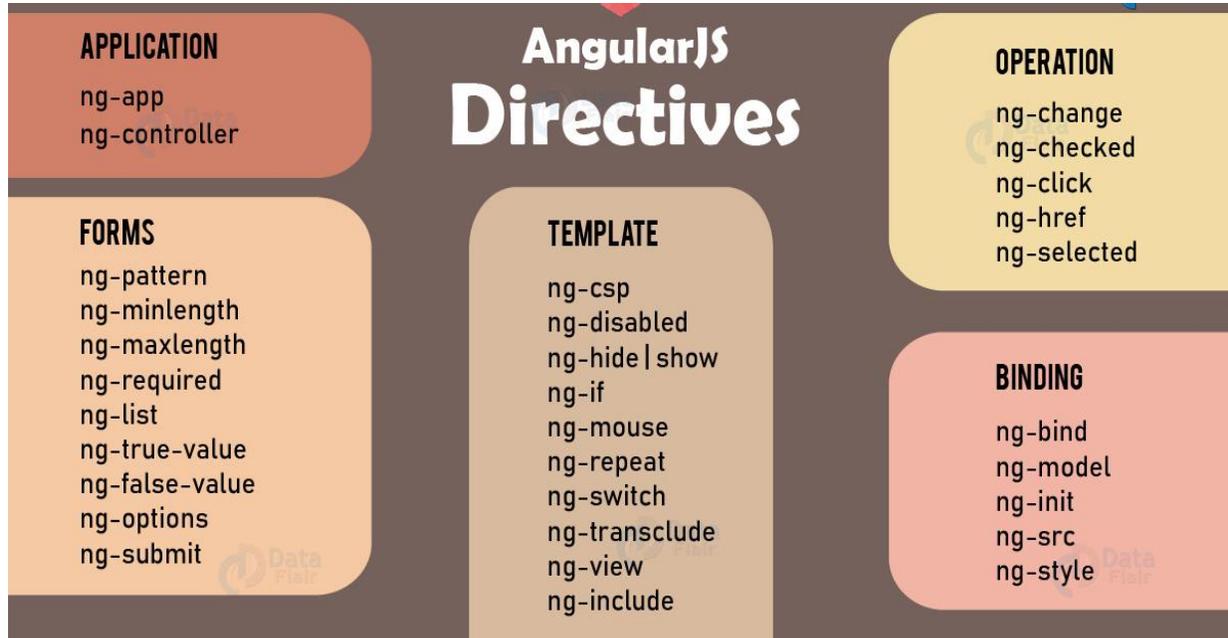
The div tag is known as Division tag. The div tag is used in HTML to make divisions of content in the web page like (text, images, header, footer, navigation bar, etc). Div tag has both open(<div>) and closing (</div>) tag and it is mandatory to close the tag. The Div is the most usable tag in web development because it helps us to separate out data in the web page and we can create a particular section for particular data or function in the web pages.

2) :- The tag is an inline container used to mark up a part of a text, or a part of a document.

The tag is easily styled by CSS or manipulated with JavaScript using the class or id attribute.

The tag is much like the <div> element, but <div> is a block-level element and is an inline element

Unit 2:- AngularJS Directives and Expressions



AngularJS Directives and Expressions:

Understanding ng attributes

- 1) The **ng-app directive** is used to show that it is an AngularJS application.
- 2) The **ng-model directive** binds the value of HTML elements to application data. It is used for inputting data, selecting data in an application.
- 3) The **ng-bind directive** is used to bind application data to the HTML view.
- 4) The **ng-init directive** initializes application data. It is used to initialize the data.



5) The **ng-repeat directive** repeats a set of HTML, a given number of times.

The set of HTML will be repeated once per item in a collection.

The collection must be an array or an object.

If you have an collection of objects, the **ng-repeat** directive is perfect for making a HTML table, displaying one table row for each object, and one table data for each object property

6) **ngShow and ngHide** are two ng directives in AngularJS used to show and hide the elements respectively. ngShow is used to show a div tab by linking it to a Boolean variable in the script. If the value of the variable is true then the item is displayed, else the item remains hidden and the vice versa happens in the case of ngHide.

7) The AngularJS **readonly directive** sets the readonly attribute on the element; if it gets that the expression inside ng-readonly is true. it used to show the only which we can not change ,it is only for read only purpose.

8) The AngularJS **ng-if directive** is used to remove the HTML elements if the expression is set to false. If the if element is set to true, a copy of the element is added in the DOM.

9) The AngularJS **ng-click directive** facilitates you to specify custom behavior when an element is clicked. So, it is responsible for the result what you get after clicking.

10) The **ng-disabled directive** sets the disabled attribute of a form field (input, select, or textarea).

11) The **ng-controller directive** adds a controller to your application.it control on the application .



AngularJS Directives

- 1) The **ng-app directive** is show that it is AngularJS application.
- 2) The **ng-model directive** for binds the value of HTML
it is used for input the data , select the data in application.
- 3) The **ng-bind directive** .it is used to binds application data to the HTML view.

program1:-

```
<html>

<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></
script>

<body>

<div ng-app="">

  <p>Input something in the input box:</p>

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

  <p ng-bind="name"></p>

</div>

</body>

</html>
```



4) The `ng-init` directive initializes application data. ,it is used to initializes the data.

program2:-

```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body>
<div ng-app="" ng-init="firstName='yogesh' ">
<p>The name is <span ng-bind="firstName"></span></p>
</div>
</body>
</html>
```

=====//=====

program3:-

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="" ng-init="firstName='John'">
<p>The name is <span ng-bind="firstName"></span></p>
</div>
</body>
</html>
```

***:-** The `` tag is an inline container used to mark up a part of a text,



or a part of a document.*/

=====//=====

AngularJS Expressions (angu 4)

AngularJS expressions are written inside double braces: **{{ expression }}**.

AngularJS will "output" data exactly where the expression is written:

```
<!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: {{ 5 + 5 }}</p>
</div></body>
</html>
```

AngularJS expressions bind AngularJS data to HTML the same way as the ng-bind directive.

```
<!DOCTYPE html> ( angu 6)
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="">
<p>Input something in the input box:</p>
<p>Name: <input type="text" ng-model="name"></p>
```



```
<p>{{name}}</p>
</div>
</body>
</html>

=====//=====

<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="" ng-init="firstName='John'">
<p>Input something in the input box:</p>
<p>Name: <input type="text" ng-model="firstName"></p>
<p>You wrote: {{ firstName }}</p>
</div>
</body>
</html>
```

Data Binding with AngularJS expressions

The `{{ firstName }}` expression, in the example above, is an AngularJS data binding expression.

Data binding in AngularJS binds AngularJS expressions with AngularJS data.

`{{ firstName }}` is bound with `ng-model="firstName"`.



In the next example two text fields are bound together with two ng-model directives:

```
<!DOCTYPE html>

<html>

<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body>

<div data-ng-app="" data-ng-init="quantity=1;price=5">

<h2>Cost Calculator</h2>

Quantity: <input type="number" ng-model="quantity">
Price: <input type="number" ng-model="price">

<p><b>Total in dollar:</b> {{quantity * price}}</p>

</div>

</body>

</html>
```

=====//=====

AngularJS Expressions with calculation

AngularJS expressions can be written inside double braces: `{{ expression }}`.

AngularJS expressions can also be written inside a directive: `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.

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

```
<!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: {{ 5 + 5 }}</p>
</div>
</body>
</html>
```

If you remove the **ng-app** directive, HTML will display the expression as it is, without solving it:

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<p>Without the ng-app directive, HTML will display the expression as it is, without solving it.</p>
<div>
<p>My first expression: {{ 5 + 5 }}</p>
</div>
</body>
</html>
```



AngularJS Numbers (gs9)

AngularJS numbers are like JavaScript numbers:

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

<div ng-app="" ng-init="quantity=6;cost=9">
<p>Total in dollar: {{ quantity * cost }}</p>
</div>

</body>
</html>
```

AngularJS Strings (gs10)

AngularJS strings are like JavaScript strings:

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

<div ng-app="" ng-init="firstName='mohan';lastName='pande'">
<p>The full name is: <span ng-bind="firstName + ' ' + lastName"></span></p>
```



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

AngularJS Arrays(gs11)

AngularJS arrays are like JavaScript arrays:

```
<!DOCTYPE html>  
<html>  
<script  
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>  
<body>  
  
<div ng-app="" ng-init="points=[1,15,19,2,40]">  
  
<p>The third result is {{ points[2] }}</p>  
  
</div>  
  
</body>  
</html>
```

AngularJS Objects(gs12)

AngularJS objects are like JavaScript objects:

```
<!DOCTYPE html>
```



```
<html>

<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body>

<div ng-app="" ng-init="person={firstName:'mohan',lastName:'pande'}">

<p>The name is {{ person.lastName }}</p>

</div>

</body>

</html>

=====//=====
```

5) The `ng-repeat` directive repeats a set of HTML, a given number of times.

The set of HTML will be repeated once per item in a collection.

The collection must be an array or an object.

If you have an collection of objects, the `ng-repeat` directive is perfect for making a HTML table, displaying one table row for each object, and one table data for each object property

```
<!DOCTYPE html>

<html>
```



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

<body>

<div ng-app="" ng-init="names=['Jani','Hege','Kai']">
  <p>Looping with ng-repeat:</p>
  <ul>
    <li ng-repeat="x in names">
      {{ x }}
    </li>
  </ul>
</div>
</body>
</html>
```

```
=====

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

<div ng-app="" ng-init="names=[
{name:'Jani',country:'Norway'},
{name:'Hege',country:'Sweden'},
{name:'Kai',country:'Denmark'}]">
```



```
<p>Looping with objects:</p>
<ul>
  <li ng-repeat="x in names">
    {{ x.name + ', ' + x.country }}</li>
</ul>
</div>
</body>
</html>
```

6) **ngShow** and **ngHide** are **two ng directives** in AngularJS used to show and hide the elements respectively. **ngShow** is used to show a div tab by linking it to a Boolean variable in the script. If the value of the variable is true then the item is displayed, else the item remains hidden and the vice versa happens in the case of ngHide

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

<body ng-app="">
  <div ng-show="true">
    <h1 style="color:green;">GeeksForGeeks</h1>
  </div>
```



</body>

</html>

7) The AngularJS `readonly` directive sets the `readonly` attribute on the element; if it gets that the expression inside `ng-readonly` is true. it used to show the only which we can not change ,it is only for read only purpose.

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:

1. `<input ng-readonly="expression"></input:>`

Parameter explanation:

expression: It specifies an expression that will set the element's `readonly` attribute if it returns true.

Let's take an example to demonstrate `ng-readonly` directive.

See this example:

```
<!DOCTYPE html>
<html>
<script src="http://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 disabled for writing.</p>
</body>
</html>
```



8) The AngularJS **ng-if directive** is used to remove the HTML elements if the expression is set to false. If the if element is set to true, a copy of the element is added in the DOM.

ngIf is different from ngShow and ngHide which show and hide the elements while ngIf completely removes and recreates the element in the DOM rather than changing its visibility.

It is supported by all HTML elements.

Syntax:

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

Parameter explanation:

expression: It specifies an expression that completely removes the element if it returns false. If it returns true, it inserts an element in the DOM instead.

Let's take an example to demonstrate ng-if directive.

See this example:

```
<!DOCTYPE html>
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body ng-app="">
Keep HTML: <input type="checkbox" ng-model="myVar" ng-init="myVar = true">
<div ng-if="myVar">
<h1>Welcome to JavaTpoint!</h1>
<p>A solution of all technologies..</p>
<hr>
</div>
<p>If you uncheck the checkbox then DIV element will be removed.</p>
<p>If you check the checkbox then DIV element will return.</p>
</body>
</html>
```



9) The AngularJS **ng-click directive** facilitates you to specify custom behavior when an element is clicked. So, it is responsible for the result what you get after clicking.

It is supported by all HTML elements.

Syntax:

1. `<element ng-click="expression"></element>`

Parameter explanation:

expression: It specifies an expression that is executed when an element is clicked.

Let's take an example to demonstrate the ng-click directive.

See this example:

Example1:

```
<!DOCTYPE html>
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body ng-app="">
<p>Click the button:</p>
<button ng-click="count = count + 1" ng-init="count=0">OK</button>
<p>The button has been clicked <strong>{{count}} </strong>times.</p>
<p><strong>Note:</strong>This example counts a value every time you click on the button and increase the value of the variable.</p>
</body>
</html>
```

ng-click directive example using function

Example2:

```
1.      <!DOCTYPE html>
2.      <html>
3.      <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
4.      <body ng-app="myApp">
5.      <div ng-controller="myCtrl">
6.          <p>Click the button to execute a function:</p>
7.          <button ng-click="myFunc()">OK</button>
8.          <p>The button has been clicked <strong>{{count}}</strong> times.</p>
9.      </div>
10.     <script>
11.     angular.module('myApp', [])
12.       .controller('myCtrl', ['$scope', function($scope) {
13.         $scope.count = 0;
14.         $scope.myFunc = function() {
15.           $scope.count++;
16.         };
17.       }]);
18.     </script>
19.     </body>
20.     </html>
```

10) The `ng-disabled` directive sets the disabled attribute of a form field (input, select, or textarea).

The form field will be disabled if the expression inside the `ng-disabled` attribute returns true.

The `ng-disabled` directive is necessary to be able to shift the value between `true` and `false`. In HTML, you cannot set the `disabled` attribute to `false` (the presence of the disabled attribute makes the element disabled, regardless of its value).

Syntax

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

Supported by `<input>`, `<select>`, and `<textarea>` elements.

```
<!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" ng-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>
```

11) The **ng-controller directive** adds a controller to your application. it control on the application . it is working with ng-model directive .

The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.

The ng-model directive can also:

- Provide type validation for application data (number, email, required).
- Provide status for application data (invalid, dirty, touched, error).
- Provide CSS classes for HTML elements.
- Bind HTML elements to HTML forms.

```
<!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="myCtrl">
Name: <input ng-model="name">
</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.name = "John Doe";
});
</script>

<p>Use the ng-model directive to bind the value of the input field to a property made in
the controller.</p>

</body>
</html>
```

Two-Way Binding

The binding goes both ways. If the user changes the value inside the input field, the AngularJS property will also change its value:

```
<!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="myCtrl">
Name: <input ng-model="name">
<h1>You entered: {{name}}</h1>
</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.name = "John Doe";
});
</script>

<p>Change the name inside the input field, and you will see the name in the header
changes accordingly.</p>

</body>
</html>
```



Unit 3- AngularJS Modules, Controller, View and Scope

AngularJS Modules

An AngularJS module defines an application.

The module is a container for the different parts of an application.

The module is a container for the application controllers.

Creating a Module

A module is created by using the AngularJS function `angular.module`

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

<script>

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

</script>
```

Adding a Controller

Add a controller to your application, and refer to the controller with the `ng-controller` directive:

```
<!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="myCtrl">
  {{ firstName + " " + lastName }}
</div>
```

```
<script>
var app = angular.module("myApp", []);
app.controller("myCtrl", function($scope) {
  $scope.firstName = "John";
  $scope.lastName = "Doe";
});
</script>
```

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

Adding a Directive

AngularJS has a set of built-in directives which you can use to add functionality to your application.

.

In addition you can use the module to add your own directives to your applications:

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



```
<body>

<div ng-app="myApp" w3-test-directive></div>

<script>
var app = angular.module("myApp", []);
app.directive("w3TestDirective", function() {
  return {
    template : "I was made in a directive constructor!"
  };
});
</script>

</body>
</html>
```

Modules and Controllers in Files

It is common in AngularJS applications to put the module and the controllers in JavaScript files.

In this example, "myApp.js" contains an application module definition, while "myCtrl.js" contains the controller:

Example

```
<!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="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script src="myApp.js"></script>
<script src="myCtrl.js"></script>

</body>
</html>
```

AngularJS Controllers

AngularJS controllers **control the data** of AngularJS applications.

AngularJS controllers are regular **JavaScript Objects**.



AngularJS applications are controlled by controllers.

The **ng-controller** directive defines the application controller.

A controller is a **JavaScript Object**, created by a standard JavaScript **object constructor**.

```
<!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="myCtrl">
```

```
First Name: <input type="text" ng-model="firstName"><br>
```

```
Last Name: <input type="text" ng-model="lastName"><br>
```

```
<br>
```

```
Full Name: {{firstName + " " + lastName}}
```

```
</div>
```

```
<script>
```

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

```
app.controller('myCtrl', function($scope) {
```

```
    $scope.firstName = "John";
```

```
    $scope.lastName = "Doe";
```

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

Application explained:

The AngularJS application is defined by **ng-app="myApp"**. The application runs inside the `<div>`.

The **ng-controller="myCtrl"** attribute is an AngularJS directive. It defines a controller.

The **myCtrl** function is a JavaScript function.

AngularJS will invoke the controller with a **\$scope** object.

In AngularJS, `$scope` is the application object (the owner of application variables and functions).

The controller creates two properties (variables) in the scope (**firstName** and **lastName**).

The **ng-model** directives bind the input fields to the controller properties (firstName and lastName).

Controller Methods

The example above demonstrated a controller object with two properties: `lastName` and `firstName`.

A controller can also have methods (variables as functions):

```
<!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="personCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{fullName()}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('personCtrl', function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
    $scope.fullName = function() {
        return $scope.firstName + " " + $scope.lastName;
    };
});
</script>

</body>
```



</html>

Controllers In External Files

In larger applications, it is common to store controllers in external files.

Just copy the code between the <script> tags into an external file named [personController.js](#):

```
<!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="personCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{fullName()}}

</div>

<script src="personController.js"></script>
```



```
</body>
```

```
</html>
```

AngularJS Scope

The scope is the binding part between the HTML (view) and the JavaScript (controller). The scope is an object with the available properties and methods.

How to Use the Scope?

When you make a controller in AngularJS, you pass the `$scope` object as an argument:

```
<!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="myCtrl">
```

```
<h1>{{carname}}</h1>
```

```
</div>
```



```
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.carname = "Volvo";
});
</script>
```

<p>The property "carname" was made in the controller, and can be referred to in the view by using the {{ }} brackets.</p>

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

When adding properties to the `$scope` object in the controller, the view (HTML) gets access to these properties.

In the view, you do not use the prefix `$scope`, you just refer to a property name, like `{{carname}}`.

Understanding the Scope

If we consider an AngularJS application to consist of:

- View, which is the HTML.
- Model, which is the data available for the current view.
- Controller, which is the JavaScript function that makes/changes/removes/controls the data.



Then the scope is the Model.

The scope is a JavaScript object with properties and methods, which are available for both the view and the controller.

```
<!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="myCtrl">
```

```
<input ng-model="name">
```

```
<h1>My name is {{name}}</h1>
```

```
</div>
```

```
<script>
```

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

```
</script>
```



<p>When you change the name in the input field, the changes will affect the model, and it will also affect the name property in the controller.</p>

</body>

</html>

Unit 4:- AngularJS Filters

AngularJS Filters

AngularJS provides filters to transform data:

- **currency** Format a number to a currency format.
- **date** Format a date to a specified format.
- **filter** Select a subset of items from an array.
- **json** Format an object to a JSON string.
- **limitTo** Limits an array/string, into a specified number of elements/characters.
- **lowercase** Format a string to lower case.
- **number** Format a number to a string.
- **orderBy** Orders an array by an expression.
- **uppercase** Format a string to upper case.

Adding Filters to Expressions

Filters can be added to expressions by using the pipe character `|`, followed by a filter.

The **uppercase** filter format strings to upper case:

<!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="personCtrl">

<p>The name is {{ lastName | uppercase }}</p>

</div>

<script>
angular.module('myApp', []).controller('personCtrl', function($scope) {
    $scope.firstName = "John",
    $scope.lastName = "Doe"
});
</script>

</body>
</html>
```

The **lowercase** filter format strings to lower case:



Example

```
<!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="personCtrl">

<p>The name is {{ lastName | lowercase }}</p>

</div>

<script>

angular.module('myApp', []).controller('personCtrl', function($scope) {

    $scope.firstName = "John",

    $scope.lastName = "Doe"

});

</script>

</body>
```



</html>

Filters are added to directives, like `ng-repeat`, by using the pipe character `|`, followed by a 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="namesCtrl">

<p>Looping with objects:</p>

<li ng-repeat="x in names | orderBy:'country'">

 {{ x.name + ', ' + x.country }}

</div>

<script>

```
angular.module('myApp', []).controller('namesCtrl', function($scope) {
```

```
    $scope.names = [
```



```
{name:'Jani',country:'Norway'},  
{name:'Carl',country:'Sweden'},  
{name:'Margareth',country:'England'},  
{name:'Hege',country:'Norway'},  
{name:'Joe',country:'Denmark'},  
{name:'Gustav',country:'Sweden'},  
{name:'Birgit',country:'Denmark'},  
{name:'Mary',country:'England'},  
{name:'Kai',country:'Norway'}  
];  
});  
</script>  
  
</body>  
</html>
```

The currency Filter

The `currency` filter formats a number as currency:

```
<!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="costCtrl">

<h1>Price: {{ price | currency }}</h1>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('costCtrl', function($scope) {

    $scope.price = 58;

});

</script>

<p>The currency filter formats a number to a currency format.</p>

</body>

</html>
```

The filter Filter

The `filter` filter selects a subset of an array.

The `filter` filter can only be used on arrays, and it returns an array containing only the matching items.



```
<!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="namesCtrl">

<ul>

  <li ng-repeat="x in names | filter : 'i'">

    {{ x }}

  </li>

</ul>

</div>

<script>

angular.module('myApp', []).controller('namesCtrl', function($scope) {

  $scope.names = [

    'Jani',

    'Carl',

    'Margareth',

    'Hege',
```



```
'Joe',  
'Gustav',  
'Birgit',  
'Mary',  
'Kai'  
];  
});  
</script>
```

<p>This example displays only the names containing the letter "i".</p>

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

Filter an Array Based on User Input

By setting the `ng-model` directive on an input field, we can use the value of the input field as an expression in a filter.

Type a letter in the input field, and the list will shrink/grow depending on the match:

```
<!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="namesCtrl">
```

```
<p>Type a letter in the input field:</p>
```

```
<p><input type="text" ng-model="test"></p>
```

```
<ul>
```

```
<li ng-repeat="x in names | filter:test">
```

```
  {{ x }}
```

```
</li>
```

```
</ul>
```

```
</div>
```

```
<script>
```

```
angular.module('myApp', []).controller('namesCtrl', function($scope) {
```

```
  $scope.names = [
```

```
    'Jani',
```

```
    'Carl',
```

```
    'Margareth',
```

```
    'Hege',
```

```
    'Joe',
```



```
'Gustav',  
  
'Birgit',  
  
'Mary',  
  
'Kai'  
  
];  
});  
</script>  
  
<p>The list will only consists of names matching the filter.</p>  
  
</body>  
</html>
```

Working with AngularJS forms

forms in AngularJS provides data-binding and validation of input controls.

Input Controls

Input controls are the HTML input elements:

- input elements
- select elements
- button elements
- textarea elements



Data-Binding

Input controls provides data-binding by using the `ng-model` directive.

```
<input type="text" ng-model="firstname">
```

The application does now have a property named `firstname`.

The `ng-model` directive binds the input controller to the rest of your application.

The property `firstname`, can be referred to in a controller:

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<script
```

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

```
<body>
```

```
<div ng-app="myApp" ng-controller="formCtrl">
```

```
<form>
```

```
  First Name: <input type="text" ng-model="firstname">
```

```
</form>
```

```
</div>
```



```
<script>

var app = angular.module('myApp', []);

app.controller('formCtrl', function($scope) {

    $scope.firstname = "John";

});

</script>

</body>

</html>

=====

<!DOCTYPE html>

<html>

<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body>

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



```
<form>
```

```
  First Name: <input type="text" ng-model="firstname">
```

```
</form>
```

```
<h1>You entered: {{firstname}}</h1>
```

```
</div>
```

<p>Change the name inside the input field, and you will see the name in the header changes accordingly.</p>

```
</body>
```

```
</html>
```

=====

Checkbox

A checkbox has the value `true` or `false`. Apply the `ng-model` directive to a checkbox, and use its value in your application.

```
<!DOCTYPE html>
```

```
<html>
```



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

<body>

<div ng-app="">

  <form>

    Check to show a header:

    <input type="checkbox" ng-model="myVar">

  </form>

  <h1 ng-show="myVar">My Header</h1>

</div>

<p>The header's ng-show attribute is set to true when the
checkbox is checked.</p>

</body>

</html>
```



=====//=====

Radiobuttons

Bind radio buttons to your application with the `ng-model` directive.

Radio buttons with the same `ng-model` can have different values, but only the selected one will be used.

```
<!DOCTYPE html>
```

```
<html>
```

```
<script
```

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

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

```
<form>
```

Pick a topic:

```
<input type="radio" ng-model="myVar" value="dogs">Dogs
```

```
<input type="radio" ng-model="myVar" value="tuts">Tutorials
```

```
<input type="radio" ng-model="myVar" value="cars">Cars
```

```
</form>
```



```
<div ng-switch="myVar">
  <div ng-switch-when="dogs">
    <h1>Dogs</h1>
    <p>Welcome to a world of dogs.</p>
  </div>
  <div ng-switch-when="tuts">
    <h1>Tutorials</h1>
    <p>Learn from examples.</p>
  </div>
  <div ng-switch-when="cars">
    <h1>Cars</h1>
    <p>Read about cars.</p>
  </div>
</div>

<p>The ng-switch directive hides and shows HTML sections
depending on the value of the radio buttons.</p>
```



</body>

</html>

=====//=====

Selectbox

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 selectbox.

```
<!DOCTYPE html>
```

```
<html>
```

```
<script
```

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

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

```
<form>
```

Select a topic:

```
<select ng-model="myVar">
```

```
<option value="">
```

```
<option value="dogs">Dogs
```



```
<option value="tuts">Tutorials
<option value="cars">Cars
</select>
</form>

<div ng-switch="myVar">
  <div ng-switch-when="dogs">
    <h1>Dogs</h1>
    <p>Welcome to a world of dogs.</p>
  </div>
  <div ng-switch-when="tuts">
    <h1>Tutorials</h1>
    <p>Learn from examples.</p>
  </div>
  <div ng-switch-when="cars">
    <h1>Cars</h1>
    <p>Read about cars.</p>
  </div>
</div>
```



</div>

</div>

<p>The ng-switch directive hides and shows HTML sections depending on the value of the dropdown list.</p>

</body>

</html>

=====//=====

Application Code

```
<!DOCTYPE html>
```

```
<html lang="en">
```

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

```
<body>
```

```
<div ng-app="myApp" ng-controller="formCtrl">
```

```
<form novalidate>
```

```
  First Name:<br>
```



```
<input type="text" ng-model="user.firstName"><br>
Last Name:<br>
<input type="text" ng-model="user.lastName">
<br><br>
<button ng-click="reset()">RESET</button>
</form>
<p>form = {{user}}</p>
<p>master = {{master}}</p>
</div>

<script>
var app = angular.module('myApp', []);
app.controller('formCtrl', function($scope) {
    $scope.master = {firstName:"John", lastName:"Doe"};
    $scope.reset = function() {
        $scope.user = angular.copy($scope.master);
    };
    $scope.reset();
});
</script>

</body>
</html>
```



Example Explained

The **ng-app** directive defines the AngularJS application.

The **ng-controller** directive defines the application controller.

The **ng-model** directive binds two input elements to the **user** object in the model.

The **formCtrl** controller sets initial values to the **master** object, and defines the **reset()** method.

The **reset()** method sets the **user** object equal to the **master** object.

The **ng-click** directive invokes the **reset()** method, only if the button is clicked.

The novalidate attribute is not needed for this application, but normally you will use it in AngularJS forms, to override standard HTML5 validation.

AngularJS Events

You can add AngularJS event listeners to your HTML elements by using one or more of these directives:

- **ng-blur**
- **ng-change**
- **ng-click**
- **ng-copy**
- **ng-cut**
- **ng-dblclick**
- **ng-focus**
- **ng-keydown**
- **ng-keypress**
- **ng-keyup**
- **ng-mousedown**
- **ng-mouseenter**
- **ng-mouseleave**



- ng-mousemove
- ng-mouseover
- ng-mouseup
- ng-paste

The event directives allows us to run AngularJS functions at certain user events.

An AngularJS event will not overwrite an HTML event, both events will be executed.

Mouse Events

Mouse events occur when the cursor moves over an element, in this order:

1. ng-mouseover
2. ng-mouseenter
3. ng-mousemove
4. ng-mouseleave

Or when a mouse button is clicked on an element, in this order:

1. ng-mousedown
2. ng-mouseup
3. ng-click

You can add mouse events on any HTML element.

```
<!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="myCtrl">
```



```
<h1 ng-mousemove="count = count + 1">Mouse Over Me!</h1>

<h2>{{ count }}</h2>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

    $scope.count = 0;

});

</script>

</body>

</html>
```

The ng-click Directive

The **ng-click** directive defines AngularJS code that will be executed when the element is being clicked.

Example

```
<!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="myCtrl">

<button ng-click="count = count + 1">Click Me!</button>

<p>{{ count }}</p>

</div>

<script>

var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.count = 0;
});
</script>

</body>

</html>

=====

You can also refer to a function:

<!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="myCtrl">

<button ng-click="myFunction()">Click Me!</button>

<p>{{ count }}</p>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

    $scope.count = 0;

    $scope.myFunction = function() {

        $scope.count++;

    }

});

</script>

</body>

</html>

=====//=====
```



Toggle, True/False

If you want to show a section of HTML code when a button is clicked, and hide when the button is clicked again, like a dropdown menu, make the button behave like a toggle switch:

```
<!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="myCtrl">

<button ng-click="myFunc()">Click Me!</button>

<div ng-show="showMe">
  <h1>Menu:</h1>
  <div>Pizza</div>
  <div>Pasta</div>
  <div>Pesce</div>
</div>

</div>

<script>
```



```
var app = angular.module('myApp', []);  
app.controller('myCtrl', function($scope) {  
    $scope.showMe = false;  
    $scope.myFunc = function() {  
        $scope.showMe = !$scope.showMe;  
    }  
});  
</script>
```

```
<p>Click the button to show/hide the menu.</p>
```

```
</body>
```

```
</html>
```

```
=====//=====
```

The showMe variable starts out as the Boolean value false.

The myFunc function sets the showMe variable to the opposite of what it is, by using the ! (not) operator.

\$event Object

You can pass the \$event object as an argument when calling the function.

The \$event object contains the browser's event object:

```
<!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="myCtrl">

<h1 ng-mousemove="myFunc($event)">Mouse Over Me!</h1>

<p>Coordinates: {{x + ', ' + y}}</p>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.myFunc = function(myE) {
        $scope.x = myE.clientX;
        $scope.y = myE.clientY;
    }
});
</script>

<p>Mouse over the heading to display the value of clientX and clientY from the
event object.</p>

</body>
```



</html>

AngularJS Form Validation

Form Validation

AngularJS offers client-side form validation.

AngularJS monitors the state of the form and input fields (input, textarea, select), and lets you notify the user about the current state.

AngularJS also holds information about whether they have been touched, or modified, or not.

You can use standard HTML5 attributes to validate input, or you can make your own validation functions.

```
<!DOCTYPE html>
```

```
<html>
```

```
<script
```

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

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

```
<p>Try writing in the input field:</p>
```

```
<form name="myForm">
```



```
<input name="myInput" ng-model="myInput" required>
</form>
```

```
<p>The input's valid state is:</p>
<h1>{{myForm.myInput.$valid}}</h1>
```

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

=====

E-mail

Use the HTML5 type `email` to specify that the value must be an e-mail:

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



<p>Try writing an E-mail address in the input field:</p>

```
<form name="myForm">
```

```
<input type="email" name="myInput" ng-model="myInput">
```

```
</form>
```

<p>The input's valid state is:</p>

```
<h1>{{myForm.myInput.$valid}}</h1>
```

<p>Note that the state of the input field is "true" before you start writing in it, even if it does not contain an e-mail address.</p>

```
</body>
```

```
</html>
```

```
=====//=====
```

Form State and Input State

AngularJS is constantly updating the state of both the form and the input fields.

Input fields have the following states:

- **\$untouched** The field has not been touched yet
- **\$touched** The field has been touched
- **\$pristine** The field has not been modified yet
- **\$dirty** The field has been modified
- **\$invalid** The field content is not valid



- `$valid` The field content is valid

They are all properties of the input field, and are either `true` or `false`.

Forms have the following states:

- `$pristine` No fields have been modified yet
- `$dirty` One or more have been modified
- `$invalid` The form content is not valid
- `$valid` The form content is valid
- `$submitted` The form is submitted

They are all properties of the form, and are either `true` or `false`.

You can use these states to show meaningful messages to the user. Example, if a field is required, and the user leaves it blank, you should give the user a warning:

```
<!DOCTYPE html>
```

```
<html>
```

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

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

```
<p>Try leaving the first input field blank:</p>
```

```
<form name="myForm">
```

```
<p>Name:
```

```
<input name="myName" ng-model="myName" required>
```



```
<span ng-show="myForm.myName.$touched &&
myForm.myName.$invalid">The name is required.</span>
```

```
</p>
```

```
<p>Address:
```

```
<input name="myAddress" ng-model="myAddress" required>
```

```
</p>
```

```
</form>
```

<p>We use the ng-show directive to only show the error message if the field has been touched AND is empty.</p>

```
</body>
```

```
</html>
```

CSS Classes

AngularJS adds CSS classes to forms and input fields depending on their states.

The following classes are added to, or removed from, input fields:

- **ng-untouched** The field has not been touched yet
- **ng-touched** The field has been touched



- `ng-pristine` The field has not been modified yet
- `ng-dirty` The field has been modified
- `ng-valid` The field content is valid
- `ng-invalid` The field content is not valid
- `ng-valid-key` One key for each validation. Example: `ng-valid-required`, useful when there are more than one thing that must be validated
- `ng-invalid-key` Example: `ng-invalid-required`

The following classes are added to, or removed from, forms:

- `ng-pristine` No fields has not been modified yet
- `ng-dirty` One or more fields has been modified
- `ng-valid` The form content is valid
- `ng-invalid` The form content is not valid
- `ng-valid-key` One key for each validation. Example: `ng-valid-required`, useful when there are more than one thing that must be validated
- `ng-invalid-key` Example: `ng-invalid-required`

The classes are removed if the value they represent is `false`.

Add styles for these classes to give your application a better and more intuitive user interface.

```
<!DOCTYPE html>
```

```
<html>
```

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

```
<style>
```

```
input.ng-invalid {  
    background-color: pink;  
}
```

```
input.ng-valid {
```



```
background-color:lightgreen;
}
</style>
<body ng-app="">

<p>Try writing in the input field:</p>

<form name="myForm">
<input name="myName" ng-model="myName" required>
</form>

<p>The input field requires content, and will therefore become
green when you write in it.</p>

</body>
</html>

=====//=====

<!DOCTYPE html>
<html>
```



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

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

```
<p>Try writing in the input field:</p>
```

```
<form name="myForm">
```

```
<input name="myInput" ng-model="myInput" required my-  
directive>
```

```
</form>
```

```
<p>The input's valid state is:</p>
```

```
<h1>{{myForm.myInput.$valid}}</h1>
```

```
<script>
```

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

```
app.directive('myDirective', function() {
```

```
    return {
```

```
        require: 'ngModel',
```

```
        link: function(scope, element, attr, mCtrl) {
```



```
function myValidation(value) {
    if (value.indexOf("e") > -1) {
        mCtrl.$setValidity('charE', true);
    } else {
        mCtrl.$setValidity('charE', false);
    }
    return value;
}
mCtrl.$parsers.push(myValidation);
}
};
});
</script>
```

<p>The input field must contain the character "e" to be consider valid.</p>

</body>

</html>



Example Explained:

In HTML, the new directive will be referred to by using the attribute `my-directive`.

In the JavaScript we start by adding a new directive named `myDirective`.

Remember, when naming a directive, you must use a camel case name, `myDirective`, but when invoking it, you must use - separated name, `my-directive`.

Then, return an object where you specify that we require `ngModel`, which is the `ngModelController`.

Make a linking function which takes some arguments, where the fourth argument, `mCtrl`, is the `ngModelController`,

Then specify a function, in this case named `myValidation`, which takes one argument, this argument is the value of the input element.

Test if the value contains the letter "e", and set the validity of the model controller to either `true` or `false`.

At last, `mCtrl.$parsers.push(myValidation);` will add the `myValidation` function to an array of other functions, which will be executed every time the input value changes.

Validation Example

```
<!DOCTYPE html>
```

```
<html>
```

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



```
<body>
```

```
<h2>Validation Example</h2>
```

```
<form ng-app="myApp" ng-controller="validateCtrl"
name="myForm" novalidate>
```

```
<p>Username:<br>
```

```
<input type="text" name="user" ng-model="user" required>
```

```
<span style="color:red" ng-show="myForm.user.$dirty &&
myForm.user.$invalid">
```

```
<span ng-show="myForm.user.$error.required">Username is
required.</span>
```

```
</span>
```

```
</p>
```

```
<p>Email:<br>
```

```
<input type="email" name="email" ng-model="email" required>
```

```
<span style="color:red" ng-show="myForm.email.$dirty &&
myForm.email.$invalid">
```



```
<span ng-show="myForm.email.$error.required">Email is
required.</span>

<span ng-show="myForm.email.$error.email">Invalid email
address.</span>

</span>

</p>

<p>
<input type="submit"
ng-disabled="myForm.user.$dirty && myForm.user.$invalid ||
myForm.email.$dirty && myForm.email.$invalid">
</p>

</form>

<script>
var app = angular.module('myApp', []);
app.controller('validateCtrl', function($scope) {
    $scope.user = 'John Doe';
    $scope.email = 'john.doe@gmail.com';
```



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

=====

Example Explained

The AngularJS directive **ng-model** binds the input elements to the model.

The model object has two properties: **user** and **email**.

Because of **ng-show**, the spans with color:red are displayed only when user or email is **\$dirty** and **\$invalid**.

AngularJS and CSS

CSS

To include W3.CSS in your AngularJS application, add the following line to the head of your document:

```
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
```

Below is a complete HTML example, with all AngularJS directives and W3.CSS classes explained.



```
<!DOCTYPE html>

<html>

<link rel="stylesheet" href="/w3css/4/w3.css">

<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.
min.js"></script>

<body ng-app="myApp" ng-controller="userCtrl">

<div class="w3-container">

<h3>Users</h3>

<table class="w3-table w3-bordered w3-striped">

<tr>

<th>Edit</th>

<th>First Name</th>

<th>Last Name</th>

</tr>

<tr ng-repeat="user in users">

<td>
```



```
<button class="w3-btn w3-ripple" ng-
click="editUser(user.id)">&#9998; Edit</button>

</td>

<td>{{ user.fName }}</td>

<td>{{ user.lName }}</td>

</tr>

</table>

<button class="w3-btn w3-green w3-ripple" ng-
click="editUser('new')">&#9998; Create New User</button>

<form ng-hide="hideform">

  <h3 ng-show="edit">Create New User:</h3>

  <h3 ng-hide="edit">Edit User:</h3>

  <label>First Name:</label>

  <input class="w3-input w3-border" type="text" ng-
model="fName" ng-disabled="!edit" placeholder="First Name">

  <label>Last Name:</label>

  <input class="w3-input w3-border" type="text" ng-
model="lName" ng-disabled="!edit" placeholder="Last Name">

  <label>Password:</label>

  <input class="w3-input w3-border" type="password" ng-
model="passw1" placeholder="Password">

  <label>Repeat:</label>
```



```
<input class="w3-input w3-border" type="password" ng-  
model="passw2" placeholder="Repeat Password">  
  
<button class="w3-btn w3-green w3-ripple" ng-disabled="error ||  
incomplete">&#10004; Save Changes</button>  
  
</form>  
  
</div>  
  
<script src= "myUsers.js"></script>  
  
</body>  
  
</html>
```

Unit 5:- Dependency Injection, Services

Dependency injection is the ability to add the functionality of components at runtime. Let's take a look at an example and the steps used to implement dependency injection.

Step 1 – Create a separate class which has the injectable decorator. The injectable decorator allows the functionality of this class to be injected and used in any Angular JS module.

```
@Injectable()  
export class classname {  
}
```

Step 2 – Next in your appComponent module or the module in which you want to use the service, you need to define it as a provider in the @Component decorator.



```
@Component ({
  providers : [classname]
})
```

Step 2 – Place the following code in the file created above.

```
import {
  Injectable
} from '@angular/core';

@Injectable()
export class appService {
  getApp(): string {
    return "Hello world";
  }
}
```

The following points need to be noted about the above program.

- The Injectable decorator is imported from the angular/core module.
- We are creating a class called appService that is decorated with the Injectable decorator.
- We are creating a simple function called getApp which returns a simple string called “Hello world”.

Step 3 – In the app.component.ts file place the following code.

```
import {
  Component
} from '@angular/core';

import {
  appService
} from './app.service';

@Component({
  selector: 'my-app',
  template: '<div>{{value}}</div>',
  providers: [appService]
})

export class AppComponent {
  value: string = "";
  constructor(private _appService: appService) { }
  ngOnInit(): void {
```



```
        this.value = this._appService.getApp ();  
    }  
}
```

The following points need to be noted about the above program.

- First, we are importing our appService module in the appComponent module.
- Then, we are registering the service as a provider in this module.
- In the constructor, we define a variable called _appService of the type appService so that it can be called anywhere in the appComponent module.
- As an example, in the ngOnInit lifecyclehook, we called the getApp function of the service and assigned the output to the value property of the AppComponent class.

Save all the code changes and refresh the browser, you will get the following output.

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.

Following are the input controls used in AngularJS forms:

- input elements
- select elements
- button elements
- textarea elements

AngularJS provides multiple events that can be associated with the HTML controls. These events are associated with the different HTML input elements.



Following is a list of events supported in AngularJS:

- ng-click
- ng-dbl-click
- ng-mousedown
- ng-mouseup
- ng-mouseenter
- ng-mouseleave
- ng-mousemove
- ng-mouseover
- ng-keydown
- ng-keyup
- ng-keypress
- ng-change

⇒ Data Binding

ng-model directive is used to provide data binding.

Let's take an example where ng-model directive binds the input controller to the rest of your application

See this example:

1. `<!DOCTYPE html>`
2. `<html lang="en">`
3. `<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>`
4. `<body>`
5. `<div ng-app="myApp" ng-controller="formCtrl">`



```
6.     <form>
7.       First Name: <input type="text" ng-model="firstname">
8.     </form>
9.   </div>
10.  <script>
11.    var app = angular.module('myApp', []);
12.    app.controller('formCtrl', function($scope) {
13.      $scope.firstname = "Ajeet";
14.    });
15.  </script>
16. </body>
17. </html>
```

You can also change the example in the following way:

See this example:

```
1.     <!DOCTYPE html>
2.     <html>
3.       <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.
4.       8/angular.min.js"></script>
5.     <body>
6.       <div ng-app="">
7.         <form>
8.           First Name: <input type="text" ng-model="firstname">
9.         </form>
10.      <h2>You entered: {{firstname}}</h2>
11.    </div>
12.    <p>Change the name inside the input field, and you will see the name
13.    in the header changes accordingly.</p>
14.  </body>
15. </html>
```

⇒ AngularJS Checkbox

A checkbox has a value true or false. The ng-model directive is used for a checkbox.

See this example:

```
1. <!DOCTYPE html>
2. <html>
3. <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/a
   angular.min.js"></script>
4. <body>
5. <div ng-app="">
6. <form>
7.   Check to show this:
8.   <input type="checkbox" ng-model="myVar">
9. </form>
10. <h1 ng-show="myVar">Checked</h1>
11. </div>
12. <p>The ng-
   show attribute is set to true when the checkbox is checked.</p>
13. </body>
14. </html>
```

⇒ AngularJS Radio Buttons

ng-model directive is used to bind radio buttons in your applications.

Let's take an example to display some text, based on the value of the selected radio buttons. In this example, we are also using ng-switch directive to hide and show HTML sections depending on the value of the radio buttons.

See this example:

```
1. <!DOCTYPE html>
2. <html>
3. <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.
   8/angular.min.js"></script>
4. <body ng-app="">
5. <form>
6.   Pick a topic:
```



```
7.      <input type="radio" ng-model="myVar" value="tuts">Tutorials
8.      <input type="radio" ng-model="myVar" value="fest">Festivals
9.      <input type="radio" ng-model="myVar" value="news">News
10.     </form>
11.     <div ng-switch="myVar">
12.       <div ng-switch-when="tuts">
13.         <h1>Tutorials</h1>
14.         <p>Welcome to the best tutorials over the net</p>
15.       </div>
16.       <div ng-switch-when="fest">
17.         <h1>Festivals</h1>
18.         <p>Most famous festivals</p>
19.       </div>
20.       <div ng-switch-when="news">
21.         <h1>News</h1>
22.         <p>Welcome to the news portal.</p>
23.       </div>
24.     </div>
25.     <p>The ng-
switch directive hides and shows HTML sections depending on the value of the
radio buttons.</p>
26.     </body>
27.     </html>
```

⇒ AngularJS Selectbox

ng-model directive is used to bind select boxes to your application.

See this example:

```
1.      <!DOCTYPE html>
2.      <html>
3.      <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.
8/angular.min.js"></script>
4.      <body ng-app="">
5.      <form>
6.      Select a topic:
7.      <select ng-model="myVar">
8.      <option value="">
```



```
9.      <option value="tuts">Tutorials
10.     <option value="fest">Festivals
11.     <option value="news">News
12.     </select>
13.     </form>
14.     <div ng-switch="myVar">
15.       <div ng-switch-when="tuts">
16.         <h1>Tutorials</h1>
17.         <p>Welcome to the best tutorials over the net.</p>
18.       </div>
19.       <div ng-switch-when="fest">
20.         <h1>Festivals</h1>
21.         <p>Most famous festivals.</p>
22.       </div>
23.       <div ng-switch-when="news">
24.         <h1>News</h1>
25.         <p>Welcome to the news portal.</p>
26.       </div>
27.     </div>
28.     <p>The ng-
        switch directive hides and shows HTML sections depending on the value of the
        radio buttons.</p>
29.     </body>
30.     </html>
```

⇒ AngularJS form example

```
1.     <!DOCTYPE html>
2.     <html>
3.       <head>
4.         <title>Angular JS Forms</title>
5.         <script src = "http://ajax.googleapis.com/ajax/libs/angularjs
        /1.3.14/angular.min.js"></script>
6.
7.       <style>
8.         table, th , td {
9.           border: 1px solid grey;
10.          border-collapse: collapse;
```



```
11.         padding: 5px;
12.     }
13.
14.     table tr:nth-child(odd) {
15.         background-color: lightpink;
16.     }
17.
18.     table tr:nth-child(even) {
19.         background-color: lightyellow;
20.     }
21. </style>
22.
23. </head>
24. <body>
25.
26.     <h2>AngularJS Sample Application</h2>
27.     <div ng-app = "mainApp" ng-
28.         controller = "studentController">
29.
30.         <form name = "studentForm" novalidate>
31.             <table border = "0">
32.                 <tr>
33.                     <td>Enter first name:</td>
34.                     <td><input name = "firstname" type = "text" ng-
35.                         model = "firstName" required>
36.                         <span style = "color:red" ng-
37.                             show = "studentForm.firstname.$dirty && studentForm.firstname.$invalid"
38.                             lid">
39.                             <span ng-
40.                                 show = "studentForm.firstname.$error.required">First Name is required.
41.                             </span>
42.                         </span>
43.                     </td>
44.                 </tr>
45.                 <tr>
46.                     <td>Enter last name: </td>
47.                     <td><input name = "lastname" type = "text" ng-
48.                         model = "lastName" required>
```



```
43.         <span style = "color:red" ng-
            show = "studentForm.lastname.$dirty && studentForm.lastname.$invalid" >
44.         <span ng-
            show = "studentForm.lastname.$error.required">Last Name is required.<
            /span>
45.         </span>
46.         </td>
47.     </tr>
48.
49.     <tr>
50.         <td>Email: </td><td><input name = "email" type =
            "email" ng-model = "email" length = "100" required>
51.         <span style = "color:red" ng-
            show = "studentForm.email.$dirty && studentForm.email.$invalid">
52.         <span ng-
            show = "studentForm.email.$error.required">Email is required.</span>
53.         <span ng-
            show = "studentForm.email.$error.email">Invalid email address.</span
            >
54.         </span>
55.         </td>
56.     </tr>
57.
58.     <tr>
59.         <td>
60.             <button ng-click = "reset()">Reset</button>
61.         </td>
62.         <td>
63.             <button ng-
            disabled = "studentForm.firstname.$dirty &&
64.                 studentForm.firstname.$invalid || studentForm.lastna
            me.$dirty &&
65.                 studentForm.lastname.$invalid || studentForm.email.
            $dirty &&
66.                 studentForm.email.$invalid" ng-
            click="submit()">Submit</button>
67.         </td>
68.     </tr>
```

```
69.
70.         </table>
71.     </form>
72. </div>
73.
74.     <script>
75.         var mainApp = angular.module("mainApp", []);
76.
77.         mainApp.controller('studentController', function($scope) {
78.             $scope.reset = function(){
79.                 $scope.firstName = "Sonoo";
80.                 $scope.lastName = "Jaiswal";
81.                 $scope.email = "sonoojaiswal@javatpoint.com";
82.             }
83.
84.             $scope.reset();
85.         });
86.     </script>
87.
88. </body>
89. </html>
```

AngularJS Form Validation

AngularJS provides client-side form validation. It checks the state of the form and input fields (input, textarea, select), and lets you notify the user about the current state.

It also holds the information about whether the input fields have been touched, or modified, or not.

Following directives are generally used to track errors in an AngularJS form:

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

⇒ AngularJS Form Validation Example



```
1.      <!DOCTYPE html>
2.      <html>
3.      <head>
4.      <title>Angular JS Forms</title>
5.      <script src = "http://ajax.googleapis.com/ajax/libs/angularjs
      /1.3.14/angular.min.js"></script>
6.
7.      <style>
8.      table, th , td {
9.      border: 1px solid grey;
10.     border-collapse: collapse;
11.     padding: 5px;
12.     }
13.
14.     table tr:nth-child(odd) {
15.     background-color: lightpink;
16.     }
17.
18.     table tr:nth-child(even) {
19.     background-color: lightyellow;
20.     }
21. </style>
22.
23. </head>
24. <body>
25.
26.     <h2>AngularJS Sample Application</h2>
27.     <div ng-app = "mainApp" ng-
      controller = "studentController">
28.
29.         <form name = "studentForm" novalidate>
30.             <table border = "0">
31.                 <tr>
32.                     <td>Enter first name:</td>
33.                     <td><input name = "firstname" type = "text" ng-
      model = "firstName" required>
34.                 <span style = "color:red" ng-
      show = "studentForm.firstname.$dirty && studentForm.firstname.$inva
      lid">
```



```
35.         <span ng-  
           show = "studentForm.firstname.$error.required">First Name is required.  
         </span>  
36.         </span>  
37.         </td>  
38.       </tr>  
39.  
40.       <tr>  
41.         <td>Enter last name: </td>  
42.         <td><input name = "lastname" type = "text" ng-  
           model = "lastName" required>  
43.         <span style = "color:red" ng-  
           show = "studentForm.lastname.$dirty && studentForm.lastname.$invali  
           d">  
44.         <span ng-  
           show = "studentForm.lastname.$error.required">Last Name is required.<  
         /span>  
45.         </span>  
46.         </td>  
47.       </tr>  
48.  
49.       <tr>  
50.         <td>Email: </td><td><input name = "email" type =  
           "email" ng-model = "email" length = "100" required>  
51.         <span style = "color:red" ng-  
           show = "studentForm.email.$dirty && studentForm.email.$invalid">  
52.         <span ng-  
           show = "studentForm.email.$error.required">Email is required.</span>  
53.         <span ng-  
           show = "studentForm.email.$error.email">Invalid email address.</span  
         >  
54.         </span>  
55.         </td>  
56.       </tr>  
57.       <tr>  
58.         <td>  
59.           <button ng-click = "reset()">Reset</button>  
60.         </td>  
61.       </td>
```



```
62.         <button ng-
           disabled = "studentForm.firstname.$dirty &&
63.             studentForm.firstname.$invalid || studentForm.lastNamena
           me.$dirty &&
64.             studentForm.lastname.$invalid || studentForm.email.
           $dirty &&
65.             studentForm.email.$invalid" ng-
           click="submit()">Submit</button>
66.         </td>
67.     </tr>
68. </table>
69. </form>
70. </div>
71. <script>
72.     var mainApp = angular.module("mainApp", []);
73.     mainApp.controller('studentController', function($scope) {
74.         $scope.reset = function(){
75.             $scope.firstName = "Sonoo";
76.             $scope.lastName = "Jaiswal";
77.             $scope.email = "sonoojaiswal@javatpoint.com";
78.         }
79.         $scope.reset();
80.     });
81. </script>
82. </body>
83. </html>
```

AngularJS AJAX

AngularJS provides a \$http service for reading data and remote servers. It is used to retrieve the desired records from a server.

AngularJS requires data in JSON format. Once the data is ready, \$http gets the data form server in the following manner:

```
1.     function employeeController($scope,$http) {
2.         r url = "data.txt";
3.
4.         $http.get(url).success( function(response) {
5.             $scope.employees = response;
```



6. });

Here the file "data.txt" is employee's records. \$http service makes an AJAX call and sets response to its property employees. This model is used to draw tables in HTML.

⇒ AngularJS AJAX Example

testAngularJS.htm

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <title>Angular JS Includes</title>
5. <style>
6.     table, th , td {
7.         border: 1px solid grey;
8.         border-collapse: collapse;
9.         padding: 5px;
10.    }
11.
12.    table tr:nth-child(odd) {
13.        background-color: #f2f2f2;
14.    }
15.
16.    table tr:nth-child(even) {
17.        background-color: #ffffff;
18.    }
19. </style>
20. </head>
21. <body>
22. <h2>AngularJS Sample Application</h2>
23. <div ng-app = "" ng-controller = "employeeController">
24.
25.     <table>
26.         <tr>
27.             <th>Name</th>
28.             <th>Age</th>
29.             <th>Salary</th>
```



```
30.         </tr>
31.
32.         <tr ng-repeat = "employee in employees">
33.             <td>{{ employee.Name }}</td>
34.             <td>{{ employee.Age }}</td>
35.             <td>{{ employee.Salary }}</td>
36.         </tr>
37.     </table>
38. </div>
39.
40.     <script>
41.         function employeeController($scope,$http) {
42.             var url = "data.txt";
43.
44.             $http.get(url).success( function(response) {
45.                 $scope.employees = response;
46.             });
47.         }
48.     </script>
49.
50.     <script src = "http://ajax.googleapis.com/ajax/libs/angularjs/1.
2.15/angular.min.js"></script>
51.
52. </body>
53. </html>
```

Here the file data.txt contains the employee's record.

"data.txt" (employee's data in JSON format)

```
1.     [
2.     {
3.         "Name" : "Mahesh Sharma",
4.         "Age" : 25,
5.         "Salary" : "20000"
6.     },
7.
8.     {
9.         "Name" : "Rohan Malik",
10.        "Age" : 20,
```



```
11.      "Salary" : "22000"
12.      },
13.
14.      {
15.      "Name" : "Robert Petro",
16.      "Age" : 45,
17.      "Salary" : "67000"
18.      },
19.
20.      {
21.      "Name" : "Jullia Roberts",
22.      "Age" : 21,
23.      "Salary" : "55000"
24.      }
```

To execute the above example, you have to deploy testAngularJS.htm and data.txt file to a web server.

Open the file testAngularJS.htm using the URL of your server in a web browser and see the result.

Output:

The result would look like this:

Name	Age	Salary
Mahesh Sharma	25	20000
Rohan Malik	20	22000
Robert Petro	45	67000
Jullia Roberts	21	55000

Table:

Name	Age	Salary
Mahesh Sharma	25	20000



Rohan Malik	20	22000
Robert Petro	45	67000
Jullia Roberts	21	55000

⇒ HTTP Service Methods

There are several shortcut methods of calling the \$http service. In the above example, .get method of the \$http service is used. Following are several other shortcut methods:

- .delete()
- .get()
- .head()
- .jsonp()
- .patch()

→ Properties

The response from the server is an object with these properties:

- .config the object used to generate the request.
- .data a string, or an object, carrying the response from the server.
- .headers a function to use to get header information.
- .status a number defining the HTTP status.
- .statusText a string defining the HTTP status.

AngularJS Animations

In AngularJS, you can build animated transition with the help of CSS. The CSS transforms the HTML elements that give you an illusion of motion.

You must include the following things to make your application ready for animations:

→ AngularJS Animate library:

- `<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular-animate.js"></script>`

→ Refer to the ngAnimate module in your application:

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

⇒ AngularJS Animation Example

```
1. <!DOCTYPE html>
2. <html>
3. <style>
4.   div {
5.     transition: all linear 0.5s;
6.     background-color: lightblue;
7.     height: 100px;
8.     width: 100%;
9.     position: relative;
10.    top: 0;
11.    left: 0;
12.  }
13.  .ng-hide {
14.    height: 0;
15.    width: 0;
16.    background-color: transparent;
17.    top: -200px;
18.    left: 200px;
19.  }
20. </style>
21. <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
22. <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular-animate.js"></script>
23. <body ng-app="ngAnimate">
24. <h1>Hide the DIV: <input type="checkbox" ng-model="myCheck"></h1>
```



25. `<div ng-hide="myCheck"></div>`
26. `</body>`
27. `</html>`

⇒ What does ngAnimate do?

The ngAnimate module does not animate HTML elements itself. It only adds and removes some pre-defined classes to make animations when ngAnimate notice certain events, like hide or show of an HTML element.

Following is a list of directives in AngularJS who add/remove classes:

- ng-show
- ng-hide
- ng-class
- ng-view
- ng-include
- ng-repeat
- ng-if
- ng-switch

AngularJS Important Questions

A list of top frequently asked **AngularJS interview questions** and answers are given below.

1) What is AngularJS?

AngularJS is an open-source JavaScript framework used to build rich and extensible web applications. It is developed by **Google** and follows the **MVC (Model View Controller)** pattern. It supports HTML as the template language and enables the developers to create extended HTML tags which will help to represent the application's content more clearly. It is easy to update and receive information from an HTML document. It also helps in writing a proper maintainable architecture which can be tested at a client-side.

2) What are the main advantages of AngularJS?

Some of the main advantages of AngularJS are given below:



- Allows us to create a single page application.
- Follows MVC design pattern.
- Predefined form validations.
- Supports animations.
- Open-source.
- Cross-browser compliant.
- Supports two-way data binding.
- Its code is unit testable.

3) What are the disadvantages of AngularJS?

There are some drawbacks of AngularJS which are given below:

- **JavaScript Dependent**
If end-user disables JavaScript, AngularJS will not work.
- **Not Secured**
It is a JavaScript-based framework, so it is not safe to authenticate the user through AngularJS only.
- **Time Consumption in Old Devices**
The browsers on old computers and mobiles are not capable or take a little more time to render pages of application and websites designed using the framework. It happens because the browser performs some supplementary tasks like **DOM (Document Object Model)** manipulation.
- **Difficult to Learn**
If you are new in AngularJS, then it will not be easy for you to deal with complex entities such as Quite layered, hierarchically and scopes. Debugging the scope is believed a tough task for many programmers.

4) Describe MVC in reference to angular.

AngularJS is based on MVC framework, where MVC stands for **Model-View-Controller**. MVC performs the following operations:

- A model is the lowest level of the pattern responsible for maintaining data.



- A controller is responsible for a view that contains the logic to manipulate that data. It is basically a software code which is used for taking control of the interactions between the Model and View.
- A view is the HTML which is responsible for displaying the data.

For example, a \$scope can be defined as a model, whereas the functions written in angular controller modifies the \$scope and HTML displays the value of scope variable.

5) What is \$scope?

A \$scope is an object that represents the application model for an Angular application.

Each AngularJS application can have only one root scope but can have multiple child scopes. **For example:**

```
1.     var app = angular.module('myApp', []);
2.     app.controller('myCtrl', function($scope) {
3.         $scope.carname = "Volvo";
4.     });
```

Some of the key characteristics of the \$scope object are given below:

- It provides observers to check for all the model changes.
- It provides the ability to propagate model changes through the application as well as outside the system to other associated components.
- Scopes can be nested in a way that they can isolate functionality and model properties.
- It provides an execution environment in which expressions are evaluated.

6) Is AngularJS dependent on JQuery?

AngularJS is a JavaScript framework with key features like models, two-way binding, directives, routing, dependency injections, unit tests, etc. On the other hand, JQuery is a JavaScript library used for DOM manipulation with no two-way binding features.



7) What IDE's are currently used for the development of AngularJS?

A term IDE stands for **Integrated Development Environment**. There are some IDE's given below which are used for the development of AngularJS:

- **Eclipse**
It is one of the most popular IDE. It supports AngularJS plugins.
- **Visual Studio**
It is an IDE from Microsoft that provides a platform to develop web apps easily and instantly.
- **WebStorm**
It is one of the most powerful IDE for modern JavaScript development. It provides an easier way to add dependencies with angular CLI.
- **Aptana**
It is a customized version of Eclipse. It is free to use.
- **Sublime Text**
It is one of the most recommendable editors for HTML, CSS, and JavaScript. It is very much compatible with AngularJS code.

8) What are the features of AngularJS?

Some important features of AngularJS are given below:

- **MVC-** In AngularJS, you just have to split your application code into MVC components, i.e., Model, View, and the Controller.
- **Validation-** It performs client-side form validation.
- **Module-** It defines an application.
- **Directive-** It specifies behavior on the DOM element.
- **Template-** It renders the dynamic view.
- **Scope-** It joins the controller with the views.
- **Expression-** It binds application data to HTML.
- **Data Binding-** It creates a two-way data-binding between the selected element and the \$ctrl.orderProp model.
- **Filter-** It provides the filter to format data.
- **Service-** It stores and shares data across the application.



- **Routing-** It is used to build a single page application.
- **Dependency Injection-** It specifies a design pattern in which components are given their dependencies instead of hard-coding them within the component.
- **Testing-** It is easy to test any of the AngularJS components through unit testing and end-to-end testing.

9) What are the directives in AngularJS?

Directives are the markers on DOM element which are used to specify behavior on that DOM element. All AngularJS directives start with the word "ng". There are many in-built directives in AngularJS such as "**ng-app**", "**ng-init**", "**ng-model**", "**ng-bind**", "**ng-repeat**" etc.

- **ng-app**
The ng-app directive is the most important directive for Angular applications. It is used to locate the beginning of an Angular application for AngularJS HTML compiler. It marks the HTML elements that Angular intends to make the root element of the application. The custom attributes use spinal-cases, whereas the corresponding directives follow the camelCase. If we do not use this directive and try to process other directives, it gives an error.
- **ng-init**
The ng-init directive is useful for initializing the data variable's inline statement of an AngularJS application. Therefore, those statements can be used in the specified blocks where we can declare them. A directive ng-init is like a local member of the ng-app directive, and it may be a single value or a group of the values. It directly supports JSON data.
- **ng-model**
The ng-model directive binds the values of HTML elements such as input, select, textarea to the application data. It provides two-way binding behavior with the model value. Sometimes, it is also used for databinding.
- **ng-bind**
The ng-bind directive is used to bind the model/variable's value to HTML controls of an AngularJS application. It can also be used with HTML tags attributes like: <p/>, and more but it does not support two-way binding. We can only check the output of the model values.
- **ng-repeat**
The ng-repeat directive is used to repeat HTML statements. It works the same as for each loop in C#, Java or PHP on a specific collection item like an array.

Let's see a simple example of AngularJS directive:

1. `<div ng-app = "" ng-init = "countries = [{locale:'en-IND',name:'India'}, {locale:'en-PAK',name:'Pakistan'}, {locale:'en-AUS',name:'Australia'}]">`
2. `<p>Enter your Name: <input type = "text" ng-model = "name"></p>`
3. `<p>Hello !</p>`
4. `<p>List of Countries with locale:</p>`
5. ``
6. `<li ng-repeat = "country in countries">`
7. `{{ 'Country: ' + country.name + ', Locale: ' + country.locale }}`
8. ``
9. ``
10. `</div>`

10) What are the controllers in AngularJS?

Controllers are JavaScript functions which are used to provide data and logic to HTML UI. It acts as an interface between Server and HTML UI. Each controller accepts \$scope as a parameter which refers to the application/module that controller is going to control. **For example:**

1. `<script>`
2. `var app = angular.module('myApp', []);`
3. `app.controller('myCtrl', function($scope) {`
4. `$scope.firstName = "Aryan";`
5. `$scope.lastName = "Khanna";`
6. `});`
7. `</script>`

11) What are the uses of controllers in AngularJS?

AngularJS controllers are used for:

- Setting the initial state of the \$scope object
- Adding behavior to the \$scope object

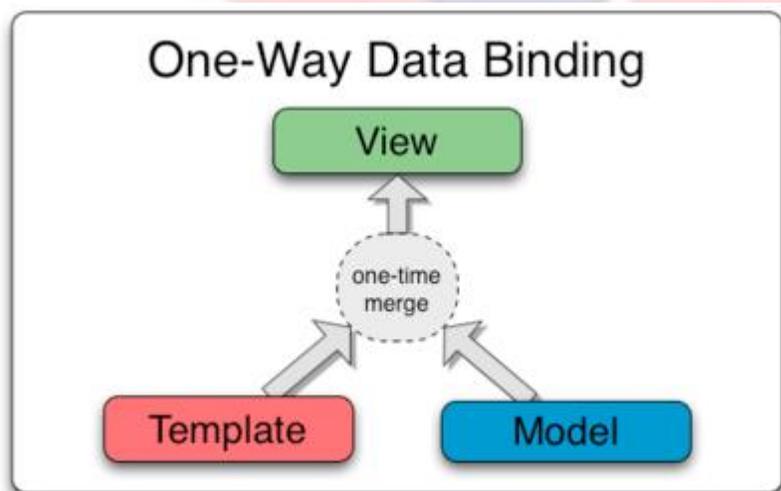
12) What is data binding in AngularJS?

Data Binding is the automatic synchronization of data between model and view. In AngularJS, it performs the automatic synchronization process between the model and view.

If the model is changed, the view reflects it automatically and vice-versa. There are two ways of data binding that AngularJS supports:

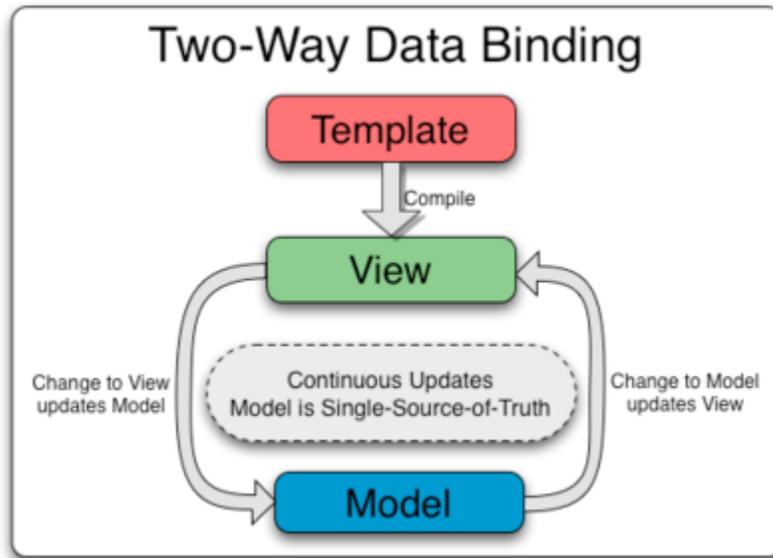
- **One Way Data Binding**

In one-way data binding, view (UI part) does not get updated automatically when the data model is changed. We need to write custom codes to make it updated every time. A directive ng-bind has one-way data binding. Here, value is taken from the data model and inserted into an HTML element.



- **Two Way Data Binding**

In two-way data binding, scope variable changes its value whenever the data model is allotted to a different value. It treats the model as the single-source-of-truth in the application. The view is a projection of the model at all time s. If the model is changed, the view reflects the change and vice versa.



13) What are the services in AngularJS?

Services are objects that can be used to store and share data across the application. AngularJS offers many built-in services, and each of them is responsible for a specific task. They are always used with the prefix \$ symbol.

Some of the important services used in any AngularJS application are as follows:

- **\$http**- It is used to make an Ajax call to get the server data.
- **\$window**- It provides a reference to a DOM object.
- **\$Location**- It provides a reference to the browser location.
- **\$timeout**- It provides a reference to the window.set timeout function.
- **\$Log**- It is used for logging.
- **\$sanitize**- It is used to avoid script injections and display raw HTML in the page.
- **\$Rootscope**- It is used for scope hierarchy manipulation.
- **\$Route**- It is used to display browser-based path in browser's URL.
- **\$Filter**- It is used for providing filter access.
- **\$resource**- It is used to work with Restful API.
- **\$document**- It is used to access the window.Document object.
- **\$exceptionHandler**- It is used for handling exceptions.
- **\$q**- It provides a promise object.
- **\$cookies**- It is used for reading, writing, and deleting the browser's cookies.
- **\$parse**- It is used to convert an AngularJS expression into a function.



- **\$cacheFactory**- It is used to evaluate the specified expression when the user changes the input.

14) What is the module in AngularJS?

A module is a container for the different parts of the application like a controller, services, filters, directives, etc. It is treated as a main() method. All the dependencies of applications are generally defined in modules only. A module is created using an angular object's module() method. **For example:**

1. `var app = angular.module('myApp', []);`

15) What is routing in AngularJS?

Routing is one of the main features of the AngularJS framework, which is useful for creating a single page application (also referred to as SPA) with multiple views. It routes the application to different pages without reloading the application. In Angular, the **ngRoute** module is used to implement Routing. The `ngView`, `$routeProvider`, `$route`, and `$routeParams` are the different components of the `ngRoute` module, which help for configuring and mapping URL to views.

16) What is a template in AngularJS?

A template consists of HTML, CSS, and AngularJS directives, which are used to render the dynamic view. It is more like a static version of a web page with some additional properties to inject and render that data at runtime. The templates are combined with information coming from model and controller.

17) What are the expressions in AngularJS?

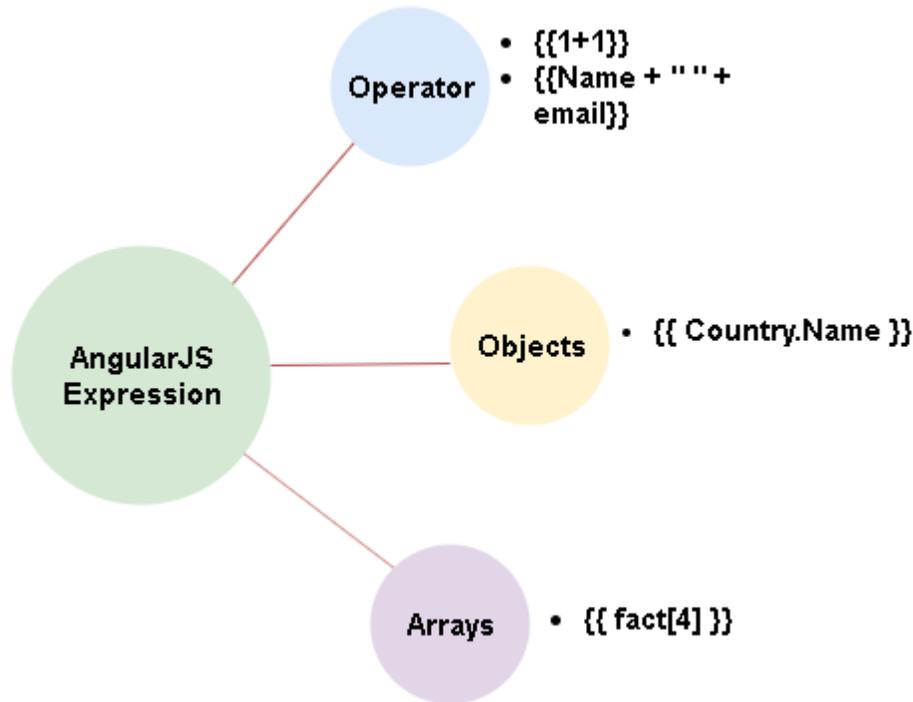
Expressions in AngularJS are the code snippets that resolve to a value. AngularJS expressions are placed inside `{{expression}}`. Expressions are included in the HTML elements.

AngularJS expressions can also contain various valid expressions similar to JavaScript expressions. We can also use the operators between numbers, including strings, literals, objects, and arrays inside the expression `{{ }}`.

For example:

1. {{1+1}}
2. {{Name + " " + email}} (string)
3. {{ Country.Name }} (object)
4. {{ fact[4] }} (array)

AngularJS supports one-time binding expressions.



18) What are the key differences between Angular expressions and JavaScript expressions?

The key differences between the Angular expressions and JavaScript expressions are given below:

Angular Expressions	JavaScript Expressions
Angular expressions do not support conditional statements, loops, and exceptions.	JavaScript expressions support conditional statements, loops, and exceptions.



Angular expressions support filters.	JavaScript expressions do not support filters.
Angular expressions can be written inside HTML.	JavaScript expressions cannot be written inside HTML.

19) What is the use of filter in AngularJS?

A filter is used to format the value of the expression to display the formatted output. AngularJS allows us to write our own filter. Filters can be added to expressions by using the pipe character |, followed by a filter. **For example:**

```
1. <div ng-app="myApp" ng-controller="personCtrl">
2. <p>The name is {{ firstName | uppercase }}</p>
3. </div>
4. <script>
5. angular.module('myApp', []).controller('personCtrl', function($scope) {
6.     $scope.firstName = "Sonoo",
7.     $scope.lastName = "Jaiswal"
8. });
9. </script>
```

Filters can be applied in view templates, controllers, services and directives. It is important to know that filters are case-sensitive. There are some built-in filters provided by AngularJS such as **Currency, Date, Filter, JSON, Limit, Lowercase, Number, Orderby, and Uppercase.**

20) What do you know about uppercase filter and lowercase filter in AngularJS?

Uppercase filters are used to convert a text to upper case text. **For example:**

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

In above example, uppercase filter is added to an expression using pipe character. It will print student name in capital letters.



On the other side, lowercase filters are used to convert a text to lower case text. **For example:**

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

It will print student name in lowercase letters.

21) Explain custom filters with an example.

We can create our own filters in AngularJS. It can be performed by associating the filter to our module. These types of filters are known as custom filters.

An example given below can be used to count the number of elements in the string by using the filter:

- ```
1. angular.module('myCountFilterApp', [])
2. .filter('count',function()
3. {
4. return(function(input)
5. {
6. var out=[];
7. out=input.split(',');
8. return out.length;
9. })
10. });
```

As per above example, if the string is "21, 34, 45" then output after applying filter will be **3**.

## 22) Explain Currency filter in AngularJS. How can we use it?

The currency filter contains the "\$" Dollar symbol as default. We can apply the following code as the html template format of Currency Filter.

1. `{{ currency_expression | currency : symbol : fractionSize}}`

We can use Currency Filter by using the following methods:



- **Default**

If we do not provide any currency symbol, then Dollar sign will be used by default as shown below:

```
<!-- by default -->
```

```
Default Currency{{amount | currency}}
```

- **User-Defined**

To use different types of currency symbols, we have to define our own symbol by applying the Hexa-Decimal code or Unicode of that Currency.

E.g., To define Indian Currency Symbol, then we have to use Unicode-value or Hexa-Decimal value.

```
Indian Currency{{amount | currency:"&# 8377"}}
```

---

## 23) What do you understand by Dependency Injection in AngularJS?

Dependency Injection (also called DI) is one of the best features of AngularJS. It is a software design pattern where objects are passed as dependencies rather than hard coding them within the component. It is useful for removing hard-coded dependencies and making dependencies configurable. To retrieve the required elements of the application that need to be configured when the module is loaded, the "config" operation uses Dependency Injection. It allows separating the concerns of different components in an application and provides a way to inject the dependent component into the client component. By using Dependency Injection, we can make components maintainable, reusable, and testable.

A simple case of dependency injection in AngularJS is shown below:

```
1. myApp.controller('myController', function ($scope, $http, $location)
2. {
3. //logic
4. });
```

Here, a controller is declared with its dependencies.

AngularJS provides the following core components which can be injected into each other as dependencies:

- Value
- Factory
- Service
- Provider



- Constant

---

## 24) What do you understand by validation of data in AngularJS?

AngularJS enriches form filling and validation. AngularJS provides client-side form validation. It checks the state of the form and input fields (input, text-area, select), and notify the user about the current state. It also holds the information about whether the input fields have been touched, or modified, or not.

There are following directives that can be used to track error:

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

Moreover, we can use **novalidate** with a form declaration to disable the browser's native form validation.

---

## 25) What do you understand by linking function? Explain its type.

Link is used for combining the directives with a scope and producing a live view. The link function is used for registering DOM listeners as well as updating the DOM. The linking function is executed as soon as the template is cloned.

There are two types of linking function:

- **Pre linking function**  
Pre-linking functions are executed before the child elements are linked. This method is not considered as a safe way for DOM transformation.
- **Post linking function**  
Post-linking functions are executed after the child elements are linked. This method is a safe way for DOM transformation.



## 26) What do you know about injector?

An injector is referred to as a service locator. It is used to receive object instances as defined by the providers, invoke methods, instantiate types, and load modules. Each Angular application consists of a single injector which helps to look upon an instance by its name.

## 27) What is the factory method in AngularJS?

Factory method is used for creating a directive. Whenever the compiler matches the directive for the first time, the factory method is invoked. Factory method is invoked using `$injector.invoke`.

### Syntax

1. `module.factory('factoryName', function);`

## 28) How will you explain the concept of hierarchy? How many scopes can an application have?

Each Angular application contains one root scope, but there can be several child scopes. The application may have multiple scopes because child controllers and some directives create new child scopes. When the new scope is formed or created, it is added as a child of the parent scope. As similar to DOM, scopes also create a hierarchical structure.

## 29) Explain how logs are maintained in AngularJS?

Logs can be maintained using **\$log** service. The main purpose of \$log service is to help in debugging and troubleshooting. It is done with the help of the following methods.

- **log()**- It writes a log message in the console.
- **info()**- It writes an information message.
- **warn()**- It writes a warning message.
- **error()**- It writes an error message.
- **debug()**- It writes a debug message.

1. `$log.error('this will displayed as an error in console')`



### 30) What is the main purpose of find index in AngularJS, and what does it return if no value is found?

Find index is used to return the position of an element. It returns the value (-1) if the requested element is not found.

1. `var index = $scope.items.findIndex(record => record.date == '2018-12-12');`

In the given code, index of the object is returned where item.date=2018-12-12.

### 31) Can we set an Angular variable from PHP session variable without sending an HTTP request?

Yes, we can perform it by injecting PHP in the required place. i.e.,

1. `$scope.name='<? = $session['name'] ?>';`

It will work only if we use PHP to render the HTML and the above JavaScript in <script> tag inside the PHP file.

### 32) What do you understand by strict conceptual escaping?

AngularJS treats all the values as untrusted/ unsecured in HTML or sensitive URL bindings. AngularJS automatically runs security checks while binding untrusted values. It throws an error if it cannot guarantee the security of the result. This type of behavior depends on contexts: HTML can be sanitized, but template URLs cannot.

To illustrate this, consider the following directive

1. `Ng-bind-html`

It renders its value directly as HTML. When there is an untrusted input, AngularJS will try to sanitize it before rendering if a sanitizer is available. We will need to mark it as trusted to bypass sanitization and render the input.

### 33) How can someone make an ajax call using AngularJS?

AngularJS contains \$https: control, which works as a service to make ajax call to read data from the server. The server creates a database call to retrieve the desired records.



AngularJS requires data in JSON format. Once the data gets ready, \$https: can be used to retrieve the data from the server in the following manner.

```
1. function studentController($scope,$https:) {
2. var url = "data.txt";
3. $https:.get(url).success(function(response) {
4. $scope.students = response;
5. });
6. }
```

### **34) What do you know about internationalization? How will you implement internationalization in AngularJS?**

Internationalization is the method for showing locale-specific information on a website. Consider a website displaying content in the English language in the United States and Danish in France.

AngularJS has inbuilt internationalization support for three types of filters:

- Currency
- Date
- Numbers

We need to incorporate the corresponding JS according to the locale of the country. By default, it is configured to handle the locale of the browser.

### **35) How will you explain deep linking in AngularJS?**

Deep linking is the method which allows us to encode the state of the application in the URL in such a way that it can be bookmarked. Then the application can further be restored from the URL to the same state.

### **36) Describe the AngularJS boot process.**

When a page is loaded into the browser, several things happen:

- HTML document file gets loaded, and evaluated by the browser. AngularJS JavaScript file gets loaded, and the angular global object is created. Next, JavaScript file which is responsible for registering the controller functions is executed.



- 
- AngularJS scans through the HTML to find AngularJS apps and views. Once the view is found, it connects that particular view to the corresponding controller function.
- 
- AngularJS executes the controller functions. It further renders the views with data from the model populated by the controller, and the page gets ready.
- 

### 37) Is it possible to have two ng-app directives for a single Angular application?

No, there can't be more than one ng-app directive for a single AngularJS application.

The ng-app directive helps AngularJS application to make sure that it is the root element. In our HTML document, we can have only one ng-app directive. If there is more than one ng-app directive, then whichever appears first will be used.

### 38) What is the syntax for creating a new date object?

The syntax for creating new date object is given below:

1. `$scope.newDate=new Date();`

### 39) Do you think that parent controller can access the methods of child controller or vice versa?

No, the parent controller cannot access the methods of child controller, but the child controller can access the methods of the parent controller.

### 40) Explain \$rootScope in AngularJS.

Every AngularJS application contains a \$rootScope, which is the top-most scope created on the DOM element. An application can contain only one \$rootScope, which will be shared among all its components. Every other scope is considered as its child scope. It can watch expressions and propagate events. By using the root scope, one can set the value in one controller and read it from the other controller.



---

## 41) What is the main purpose of \$routeProvider in AngularJS?

\$routeProvider is one of the important services which set the configuration of URLs. It further maps them with the corresponding HTML pages or ng-templates and attaches a controller with the same.

---

## 42) How will you explain Auto Bootstrap Process in AngularJS?

AngularJS initializes automatically upon the "**DOMContentLoaded**" event. It also initializes when the browser downloads the Angular.js script and document.readyState is set to 'complete' at the same time. AngularJS looks for an ng-app directive which is the root of Angular application compilation process.

If the directive 'ng-app' is found, then AngularJS will perform the following steps:

- It will load the module which is associated with the directive.
- It will create the application injector.
- It will compile the DOM starting from the ng-app root element.

This process is known as Auto-bootstrapping.

---

## 43) How will you explain Manual Bootstrap Process in AngularJS?

Sometimes, we may need to manually initialize the Angular application to have more control over the initialization process. We can perform such task using **angular.bootstrap()** function within **angular.element(document).ready()** function. AngularJS uses this function when the DOM is ready for manipulation.

The angular.bootstrap() function uses two parameters, the document, and the module name injector.



## 44) What do you understand by \$watch?

In angularJS, \$watch() function is used to watch the changes of variable in \$scope object. Generally, the \$watch() function is created internally to handle variable changes in the application.

If there is a need to create custom watch for some specific action then it's better to use \$scope.watch function. The **\$scope.watch()** function is used to create a watch of some variable. When we register a watch, we pass two functions as parameters to the \$watch() function:

- A value function
- A listener function

An example is given below:

```
1. $scope.$watch(function() {},
2. function() {}
3.);
```

Here, the first function is the value function and the second function is the listener function.

## 45) What are the different types of directives available in AngularJS?

AngularJS provides support for creating custom directives for the following type of elements:

- **Element Directive**  
Element directives are activated when a matching element is encountered.
- **Attribute**  
Attribute directives are activated when a matching attribute is encountered.
- **CSS**  
CSS directives are activated when a matching CSS style is encountered.
- **Comment**  
Comment directives are activated when a matching comment is encountered.



## 46) Explain the compilation process of AngularJS?

Angular's HTML compiler allows us to teach the browser, new HTML syntax. It also allows the developer to attach new behavior or attributes to any HTML element known as directives. AngularJS compilation process automatically takes place in the web browser. It does not contain any server-side or pre-compilation procedure.

AngularJS uses `<$compiler>` service for the compilation process of an Angular HTML page. Its compilation process starts after the HTML page (static DOM) is completely loaded.

It occurs in two phases:

- **Compile**  
It checks into the entire DOM and collects all of the directives.
- **Link**  
It connects the directives with a scope and produces a live view.

The concept of compile and link has been added from C language. The code is compiled and then linked.

## 47) What is the Global API in AngularJS?

Global API is the combination of global JavaScript function, which is used to perform tasks such as comparing objects, iterating objects, and converting the data.

There are a few common API functions like:

- **angular.lowercase**  
It is used to convert a string to lowercase string.
- **angular.uppercase**  
It is used to convert a string to uppercase string.
- **angular.IsString**  
It returns true if the current reference is a string.
- **angular.IsNumber**  
It returns true if the current reference is a number.

## 48) Is AngularJS well-suited with all browsers?

Yes, AngularJS is supported with all the browsers like Safari, Chrome, Mozilla, Opera, and Internet Explorer, etc. It is also companionable with mobile browsers.



## 49) "How are AngularJS prefixes \$ and \$\$ used?

\$\$ prefix in AngularJS is used as a private variable, as it is responsible for preventing accidental code collision with the user code.

Whereas, \$ prefix is used to define angular core functionalities such as variable, parameter, property or method, etc.

## 50) How can someone set, get, and clear cookies in AngularJS?

AngularJS has a module known as ngCookies. Before we inject ngCookies, we should include angular-cookies.js into the application.

- o **Set Cookies**

We can use the put method to set cookies in a key-value format.

1. `$cookies.put("username", $scope.username);`

- o **Get Cookies**

We can use the get method to get cookies.

1. `$cookies.get('username');`

- o **Clear Cookies**

We can use the remove method to remove or clear cookies.

1. `$cookies.remove('username');`

=====`//end//`=====



DACC

## Bibliography

| Learning Resources: |                 |                                                                                                                                                                                         |
|---------------------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1                   | Reference Books | 1) Black Book - Web Technology : Kogent Learning Solution<br>2) Complete Reference with Angular js:- Asim Hussain.<br>3) Mastering Web Application Development with AngularJS by Pawel. |
| 2                   | Websites        | a) <a href="http://www.W3School.com">http://www.W3School.com</a><br>b) <a href="http://www.tutorialpoint.com">http://www.tutorialpoint.com</a>                                          |