**Angular JS**

Angular JS is an open source JavaScript framework by Google to build web applications. It can be freely used, changed and shared by anyone.

AngularJS such as mvc, expressions, directives, controllers, modules, scopes, filters, dom, forms, ajax, validation, services, animation, dependency injection, views, w3.css etc.

What is AngularJS?

Angular JS is an open source JavaScript framework that is used to build web applications. It can be freely used, changed and shared by anyone. Angular Js is developed by Google.It is an excellent framework for building single phase applications and line of business applications.

Advantages of AngularJS:

1. Dependency Injection
2. Two way data binding
3. Testing
4. Model View Controller

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.
2. ****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.
3. ****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.

Event Controller View Model

1. AngularJS applications are a mix of HTML and JavaScript. The first thing you need is an HTML page.
2. Second, you need to include the AngularJS JavaScript file in the HTML page so we can use AngularJS:

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js"></script>

</head>

<body>

.

.

</body>

</html>

AngularJS directives are HTML attributes with an ****ng**** prefix.

The ****ng-app**** directive tells AngularJS that the <div> element is the "owner" of an AngularJS ****application****.

The ****ng-model**** directive binds the value of the input field to the application variable ****name****.

The ****ng-bind**** directive binds the content of the <p> element to the application variable ****name****.

The ****ng-init**** directive initializes AngularJS application variables.

<html lang="en">

<head>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js"></script>

</head>

<body ng-app="myapp">

<div ng-controller="HelloController" >

<h2>Hello {{helloTo.title}} !</h2>

</div>

<script>

angular.module("myapp", [])

.controller("HelloController", function($scope) {

$scope.helloTo = {};

$scope.helloTo.title = "World, AngularJS";

} );

</script>

</body>

</html>

ng-app = used to initialize an AngularJS application/ module.

ng-controller = Controllers in AngularJS are responsible for managing the data and logic for a specific part of the webpage.

AngularJS interpolation syntax ({{}})

$scope is an object that holds data accessible to both the controller and the associated HTML template.

$scope.helloTo is an empty object where you can store data related to a specific part of your template (in this case, the "HelloController").

$scope.helloTo.title is a property within $scope.helloTo that holds the text "World, AngularJS."

<script>: This script tag is where you define your AngularJS application and controller.

Modules in AngularJS are used to organize and configure different parts of your application.

**MODEL CODE:**

<!DOCTYPE html>

<html lang = "en">

<head>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js">

</script>

</head>

<body ng-app="demo">

<div ng-controller ="worldDemo" >

<h2> Hello World! {{helloTo.koki}}</h2>

</div>

<script>

angular.module("demo", [])

.controller("worldDemo",function($scope){

$scope.helloTo={};

$scope.helloTo.koki="Angular is the Best";

} );

</script>

</body>

</html>

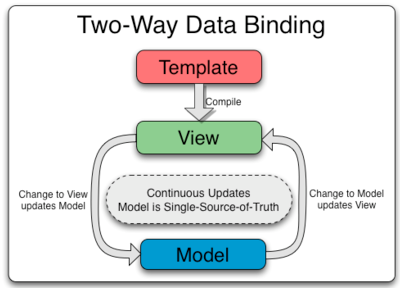
Data binding is a very useful and powerful feature used in software development technologies. It acts as a bridge between the view and business logic of the application.

## **One-Way Data Binding**

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

**Two-Way Data Binding**

Data-binding in Angular apps is the automatic synchronization of data between the model and view components.



<!DOCTYPE html>

<html>

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

<body>

<div data-ng-app="" data-ng-init="quantity=1;price=20">

<h2>Cost Calculator</h2>

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

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

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

</div>

</body>

</html>

**Angular JS -Expression**

Expressions are used to bind application data to HTML. AngularJS resolves the expression, and return the result exactly where the expression is written.

Expressions are written inside double braces {{expression}}.They can also be written inside a directive:

ng-bind="expression".

CODE:

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>

<p>Change the value of the input field:</p>

<div ng-app="" ng-init="myCol='pink'">

<input ng-style="background-color:{{myCol}}" ng-model="myCol" value="{{myCol}}">

</div>

<p>AngularJS resolves the expression and returns the result.</p>

<p>The background color of the input box will be whatever you write in the input field.</p>

</body>

</html>

Need of ng-bind :It is often preferred in scenarios where you want to bind data dynamically, especially when dealing with complex expressions or when you want to ensure that the content inside the element gets updated properly. It provides a cleaner separation between the data binding logic and the template, and it can be more explicit in terms of indicating what should be bound.

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

AngularJS Directives List

AnglarJS directives are used to add functionality to your application. You can also add your own directives for your applications.

The **ng-model** directive can also:

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

Following is a list of AngularJS directives:

**Directive Description**

ng-app It defines the root element of an application.

ng-bind It binds the content of an html element to application data.

ng-bind-html It binds the inner HTML of an HTML element to application data, and also removes dangerous code from the html string.

ng-bind-template It specifies that the text content should be replaced with a template.

ng-blur It specifies a behavior on blur events.

ng-change It specifies an expression to evaluate when content is being changed by the user.

ng-checked It specifies if an element is checked or not.

ng-class It specifies css classes on html elements.

ng-class-even It is same as ng-class, but will only take effect on even rows.

ng-class-odd It is same as ng-class, but will only take effect on odd rows.

ng-click It specifies an expression to evaluate when an element is being clicked.

ng-cloak It prevents flickering when your application is being loaded.

ng-controller It defines the controller object for an application.

ng-copy It specifies a behavior on copy events.

ng-csp It changes the content security policy.

ng-cut It specifies a behavior on cut events.

ng-dblclick It specifies a behavior on double-click events.

ng-focus It specifies a behavior on focus events.

ng-hide It hides or shows html elements.

ng-href It specifies a URL for the <a> element.

ng-if It removes the html element if a condition is false.

ng-include It includes html in an application.

ng-init It defines initial values for an application.

ng-jq It specifies that the application must use a library, like jQuery.

ng-keydown It specifies a behavior on keydown events.

ng-keypress It specifies a behavior on keypress events.

ng-keyup It specifies a behavior on keyup events.

ng-list It converts text into a list (array).

ng-open It specifies the open attribute of an element.

ng-options It specifies <options> in a <select> list.

ng-paste It specifies a behavior on paste events.

ng-pluralize It specifies a message to display according to en-us localization rules.

ng-readonly It specifies the readonly attribute of an element.

ng-required It specifies the required attribute of an element.

ng-selected It specifies the selected attribute of an element.

ng-show It shows or hides html elements.

ng-src It specifies the src attribute for the <img> element.

ng-srcset It specifies the srcset attribute for the <img> element.

ng-style It specifies the style attribute for an element.

ng-submit It specifies expressions to run on onsubmit events.

ng-switch It specifies a condition that will be used to show/hide child elements.

ng-transclude It specifies a point to insert transcluded elements.

ng-value It specifies the value of an input element.

ng-disabled It specifies if an element is disabled or not.

ng-form It specifies an html form to inherit controls from.

ng-model It binds the value of html controls to application data.

ng-model-options It specifies how updates in the model are done.

ng-mousedown It specifies a behavior on mousedown events.

ng-mouseenter It specifies a behavior on mouseenter events.

ng-mouseleave It specifies a behavior on mouseleave events.

ng-mousemove It specifies a behavior on mousemove events.

ng-mouseover It specifies a behavior on mouseover events.

ng-mouseup It specifies a behavior on mouseup events.

ng-non-bindable It specifies that no data binding can happen in this element, or it's children.

ng-repeat It defines a template for each data in a collection.

**As attribute**

<element ng-bind="expression"></element>

**As CSS Class**

<element class="ng-bind: expression"></element>

var app = angular.module('myApp', []);: This line creates an AngularJS module named "myApp." The empty array ([]) is used to define dependencies, but in this case, there are none.

app.controller('myCtrl', function($scope) { ... });: This code defines an AngularJS controller named "myCtrl." The controller function is responsible for setting up the data and logic for the application. In this case, it assigns the value "Welcome to JavaTpoint" to the $scope.myVar variable.

****expression:**** It specifies a variable or an expression to evaluate.

The angular-sanitize module provides functionality for safely rendering and manipulating HTML content within AngularJS applications.

**AngularJs Controller:**

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/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 = "Aryan";

$scope.lastName = "Khanna";

});

</script>

</body>

</html>

**AngularJs Controller With Function**

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/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 = "Aryan";

$scope.lastName = "Khanna";

$scope.fullName = function() {

return $scope.firstName + " " + $scope.lastName;

};

});

</script>

</body>

</html>

**AngularJs Scopes**

The Scope is an object that is specified as a binding part between the HTML (view) and the JavaScript (controller). It plays a role of joining controller with the views. It is available for both the view and the controller.

<!DOCTYPE html>

<html>

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

//its give output as carname

**Constants Example:**

<!DOCTYPE html>

<html>

<head>

<title>AngularJS Dependency Injection</title>

</head>

<body>

<h2>AngularJS Sample Application</h2>

<div ng-app = "mainApp" ng-controller = "CalcController">

<p>Enter a number: <input type = "number" ng-model = "number" /></p>

<button ng-click = "square()">X<sup>2</sup></button>

<p>Result: {{result}}</p>

</div>

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

<script>

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

mainApp.config(function($provide) {

$provide.provider('MathService', function() {

this.$get = function() {

var factory = {};

factory.multiply = function(a, b) {

return a \* b;

}

return factory;

};

});

});

mainApp.value("defaultInput", 10);

mainApp.factory('MathService', function() {

var factory = {};

factory.multiply = function(a, b) {

return a \* b;

}

return factory;

});

mainApp.service('CalcService', function(MathService){

this.square = function(a) {

return MathService.multiply(a,a);

}

});

mainApp.controller('CalcController', function($scope, CalcService, defaultInput) {

$scope.number = defaultInput;

$scope.result = CalcService.square($scope.number);

$scope.square = function() {

$scope.result = CalcService.square($scope.number);

}

});

</script>

</body>

</html>

**FILTERS**

**UPPER CASE FILTERS**

<!DOCTYPE html>

<html>

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

<body>

<div ng-app="myApp" ng-controller="personCtrl">

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

</div>

<script>

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

$scope.firstName = "kokila",

$scope.lastName = "Jaiswal"

});

</script>

</body>

</html>

OUTPUT: KOKILA

**ANGULAR CSS STYLE TAGS**

<!DOCTYPE html>

<html>

<head>

<title>Angular JS Table</title>

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

<style>

table, th , td {

border: 1px solid grey;

border-collapse: collapse;

padding: 5px;

}

table tr:nth-child(odd) {

background-color: #f2f2f2;

}

table tr:nth-child(even) {

background-color: #ffffff;

}

</style>

</head>

<body>

<h2>AngularJS Sample Application</h2>

<div ng-app = "mainApp" ng-controller = "studentController">

<table border = "3">

<tr>

<td>Enter first name:</td>

<td><input type = "text" ng-model = "student.firstName"></td>

</tr>

<tr>

<td>Enter last name: </td>

<td>

<input type = "text" ng-model = "student.lastName">

</td>

</tr>

<tr>

<td>Name: </td>

<td>{{student.fullName()}}</td>

</tr>

<tr>

<td>Subject:</td>

<td>

<table>

<tr>

<th>Name</th>

<th>Marks</th>

</tr>

<tr ng-repeat = "subject in student.subjects">

<td>{{ subject.name }}</td>

<td>{{ subject.marks }}</td>

</tr>

</table>

</td>

</tr>

</table>

</div>

<script>

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

mainApp.controller('studentController', function($scope) {

$scope.student = {

firstName: "Rahul",

lastName: "Rai",

fees:500,

subjects:[

{name:'Physics',marks:850},

{name:'Chemistry',marks:80},

{name:'Math',marks:90},

{name:'English',marks:80},

{name:'Hindi',marks:70}

],

fullName: function() {

var studentObject;

studentObject = $scope.student;

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

}

};

});

</script>

</body>

</html>

**SELECTION BASED**

<!DOCTYPE html>

<html>

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

<body>

<div ng-app="myApp" ng-controller="myCtrl">

<p>Select a car:</p>

<select ng-model="selectedCar" ng-options="x for (x, y) in cars">

</select>

<h1>You selected: {{selectedCar.brand}}</h1>

<h2>Model: {{selectedCar.model}}</h2>

<h3>Color: {{selectedCar.color}}</h3>

</div>

<script>

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

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

$scope.cars = {

car01 : {brand : "Ford", model : "Mustang", color : "red"},

car02 : {brand : "Honda", model : "city", color : "white"},

car03 : {brand : "Volvo", model : "XC90", color : "black"},

car04 : {brand : "Hundai", model : "Creta", color : "gray"}

}

});

</script>

</body>

</html>

The AngularJS HTML DOM is a key aspect of how AngularJS applications work. It allows you to create dynamic and interactive web applications by providing a structured and programmable representation of the web page's content. AngularJS simplifies DOM manipulation and interaction through its directives and data-binding capabilities, making it easier to develop rich web applications.

**BUTTON DIFFERENCE**

<!DOCTYPE html>

<html>

<head>

<title>AngularJS HTML DOM</title>

</head>

<body>

<h2>AngularJS Sample Application</h2>

<div ng-app = "">

<table border = "0">

<tr>

<td><input type = "checkbox" ng-model = "enableDisableButton">Disable Button</td>

<td><button ng-disabled = "enableDisableButton">Click Me!</button></td>

</tr>

<tr>

<td><input type = "checkbox" ng-model = "showHide1">Show Button</td>

<td><button ng-show = "showHide1">Click Me!</button></td>

</tr>

<tr>

<td><input type = "checkbox" ng-model = "showHide2">Hide Button</td>

<td><button ng-hide = "showHide2">Click Me!</button></td>

</tr>

<tr>

<td><p>Total click: {{ clickCounter }}</p></td>

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

</tr>

</table>

</div>

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

</body>

</html>

**Event Supported In AngularJs**

1. ng-click
2. ng-dbl-click
3. ng-mousedown
4. ng-mouseup
5. ng-mouseenter
6. ng-mouseleave
7. ng-mousemove
8. ng-mouseover
9. ng-keydown
10. ng-keyup
11. ng-keypress
12. ng-change

**$dirty**

In AngularJS, the $dirty property is used to determine whether a form control (such as an input field) has been interacted with or modified by the user. It is a part of AngularJS's form validation features and is typically used with form elements and the ng-model directive.

Here's how $dirty works:When a form control is initially loaded, it is considered "pristine," meaning it hasn't been modified by the user.As soon as the user interacts with the control by typing into an input field, checking a checkbox, or making a selection in a dropdown, the control becomes "dirty." This means it has been touched or modified by the user.$dirty is a boolean property associated with the form control through AngularJS's form controller. It becomes true when the control becomes dirty and false when it's pristine.

**AngularJS controllers are used for:**

1. Setting the initial state of the $scope object
2. Adding behavior to the $scope object