AnGULAR JS

AngularJS is a client side JavaScript MVC framework to develop a dynamic web application.

AngularJS changes static HTML to dynamic HTML.

AngularJS can be used to create Single Page Applications.

AngularJS extends HTML attributes with **Directives**, and binds data to HTML with **Expressions**.

AngularJS website - [https://angularjs.org](https://angularjs.org/)

## AngularJS Extends HTML

AngularJS extends HTML with **ng-directives**.

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

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

The **ng-bind** directive binds application data to the HTML view.

Most of the directives in AngularJS are starting with **ng**. It stands for Angular.

Example: First AngularJS Application

<!DOCTYPE html>  
<html>  
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>  
<body>  
  
<div ng-app="">  
  <p>Name: <input type="text" ng-model="name"></p>  
  <p ng-bind="name"></p>  
</div>  
  
</body>  
</html>

## Expression:

An expression is like JavaScript code which is usually wrapped inside double curly braces such as {{ expression }}. AngularJS framework evaluates the expression and produces a result. In the above example, {{ Num1 \* Num2}} will simply display the product of Num1 and Num2.

<!DOCTYPE html>  
<html>  
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>  
<body>  
  
<div ng-app="">  
  <p>My first expression: {{ 5 + 5 }}</p>  
</div>  
  
</body>  
</html>

**ng-app with Module name:**

The ng-app directive can also specify an application module name. This application module separates different parts of your application such as controllers, services, filters etc.

Example: ng-app with App Module

<!DOCTYPE html>

<html>

<head>

<title>ng-app Directive</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>  
</head>

<body>

<div ng-app="myAngularApp">

<h1>Hi</h1>

</div>  
  
<script>

// AngularJS modules define applications:

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

</script>

</body>

</html>

**Manual Bootstrap:**

We have learned that the ng-app directive auto initializes an AngularJS framework. However, we can also initialize AngularJS manually without using ng-app directive.

The following example demonstrates manual initialization of Angular.

Example: Manual Bootstrap

<!DOCTYPE html>

<html >

<head>

<title>Angular Bootstrap</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>  
</head>

<body>

<div>

<h1>Hi</h1>

</div>

<script>

angular.element(document).ready(function () {

angular.bootstrap(document);

});

</script>

</body>

</html>

**Adding a Controller**

<!DOCTYPE html>

<html >

<head>

<title>Angular Bootstrap</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>  
</head>

<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>

## AngularJS Directives:

We used directives in our first AngularJS application section. Here, we will learn directives in detail.

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

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

The following table lists the important built-in AngularJS directives.

Directive Description

ng-app Auto bootstrap AngularJS application.

ng-init Initializes AngularJS variables

ng-model Binds HTML control's value to a property on the $scope object.

ng-controller Attaches the controller of MVC to the view.

ng-bind Replaces the value of HTML control with the value of specified AngularJS expression.

ng-repeat Repeats HTML template once per each item in the specified collection.

ng-show Display HTML element based on the value of the specified expression.

ng-readonly Makes HTML element read-only based on the value of the specified expression.

ng-disabled Sets the disable attribute on the HTML element if specified expression evaluates to true.

ng-if Removes or recreates HTML element based on an expression.

ng-click Specifies custom behavior when an element is clicked.

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

<div ng-app="" ng-init="firstName='John'">  
  
<p>The name is <span ng-bind="firstName"></span></p>  
  
</div>

The ng-repeat directive repeats an HTML element:

<div ng-app="" ng-init="names=['Jani','Hege','Kai']">  
  <ul>  
    <li ng-repeat="x in names">  
      {{ x }}  
    </li>  
  </ul>  
</div>

The ng-repeat directive used on an array of objects:

<div ng-app="" ng-init="names=[  
{name:'Jani',country:'Norway'},  
{name:'Hege',country:'Sweden'},  
{name:'Kai',country:'Denmark'}]">  
  
<ul>  
  <li ng-repeat="x in names">  
    {{ x.name + ', ' + x.country }}  
  </li>  
</ul>  
  
</div>

**ng-if:**

The ng-if directive creates or removes an HTML element based on the Boolean value returned from the specified expression. If an expression returns true then it recreates an element otherwise removes an element from the HTML document.

Keep HTML: <input type="checkbox" ng-model="myVar" ng-init="myVar = true">  
<div ng-if="myVar">  
<h1>Welcome</h1>  
<p>Welcome to my home.</p>  
<hr>  
</div>

<p>The DIV element will be removed when the checkbox is not checked.</p>

<p>The DIV element will return, if you check the checkbox.</p>

AngularJS **ng-show** Directive

The ng-show directive shows the specified HTML element if the expression evaluates to true, otherwise the HTML element is hidden.

Show a section when a checkbox is checked:

Show HTML: <input type="checkbox" ng-model="myVar">  
<div ng-show="myVar">  
<h1>Welcome</h1>  
<p>Welcome to my home.</p>  
</div>

AngularJS **ng-hide** Directive

The ng-hide directive hides the HTML element if the expression evaluates to true.

Hide a section when a checkbox is checked:

Hide HTML: <input type="checkbox" ng-model="myVar">  
<div ng-hide="myVar">  
<h1>Welcome</h1>  
<p>Welcome to my home.</p>  
</div>

**Creating custom directive**

A directive is, thus, implemented as a function, which returns an object, called Data Definition Object (DDO), that configures the directive’s behavior and template. **restrict, template,**etc., are the fields of this object.

**restrict**: defines the type of HTML element which can act as a trigger for the directive

You can invoke a directive by using:

* E: Element name <my-directive ></my-directive>
* A: Attribute <div my-directive></div>
* C: Class <div class=”my-directive”></div>
* M: Comment <!-- directive: my-directive -->

**transclude**: specifies whether to transfer and include the original inner content of the directive’s HTML markup (‘Movie Title :’) in the destination markup (which is defined in the template).

**template**: specifies the HTML content that should be added to the HTML result of the directive.

<!DOCTYPE html>

<html >

<head>

<title>Angular custom directive</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>  
</head>

<body>

<div ng-app=”myApp”>

<my-directive></my-directive>

</div>

<script>

var app = angular.module(‘myApp, []);

app.directive(‘myDirective’, function(){

restrict: ‘E’,

return {

template: ‘<h1>Hi this is custom directive</h1>’;

}

});

</script>

</body>

</html>

Eg. 2

<!DOCTYPE html>

<html >

<head>

<title>Angular custom directive</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>  
</head>

<body>

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

    <div ng-repeat="movie in movies">

<my-movie title="{{movie}}">

    Movie Title :

</my-movie>

    </div>

</div>

<script>

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

myApp.controller('myController', function($scope) {

    $scope.movies = ['Ice Age', 'Frozen','Aladdin', 'Tangled', 'Cars'];

});

myApp.directive('myMovie', function() {

  return {

    restrict: 'E',

    transclude: 'true',

    template: '<span ng-transclude></span>',

    link: function(scope, element, attr){

        element.append("<strong>"+attr.title+"</strong>");

        if(attr.title === 'Ice Age'){

            element.append("<br> say hi to Manny!");

        }

        else {

            element.append("<br> all hail the Snow Queen!");

        }

      }

  };

});

</script>

</body>

</html>

**Two-way Binding**

Data binding in AngularJS is the synchronization between the model and the view.

When data in the model changes, the view reflects the change, and when data in the view changes, the model is updated as well. This happens immediately and automatically, which makes sure that the model and the view is updated at all times.

<div ng-app="myApp" ng-controller="myCtrl">  
    Name: <input ng-model="firstname">  
     <h1>{{firstname}}</h1>  
</div>  
  
<script>

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

</script>