AngularJS

AngularJS extends HTML with new attributes. AngularJS is perfect for Single Page Applications (SPAs).

Basics of AngularJS: directives, expressions, filters, modules, and controllers. Events, DOM, Forms, Input, Validation, Http, and more.

Before you study AngularJS, you should have a basic understanding of:

HTML

CSS

JavaScript

AngularJS is a **JavaScript framework**. It can be added to an HTML page with a <**script**> tag.

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

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

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.

```
<!DOCTYPE html>

<html>

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

Example explained:

AngularJS starts automatically when the web page has loaded.

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.

AngularJS Directives

As you have already seen, AngularJS directives are HTML attributes with an **ng** prefix.

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>
```

You can use **data-ng-**, instead of **ng-**, if you want to make your page HTML valid.

AngularJS Expressions

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

```
<!DOCTYPE html>

<html>

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

<body>

<div ng-app="">

  <p>My first expression: {{ 5 + 5 }}</p>

</div>

</body>

</html>
```

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

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

```
<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>
```

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

Change the color of the input box below, by changing its value:

```
<div ng-app="" ng-init="myCol='lightblue'">
<input style="background-color:{{myCol}}" ng-model="myCol">
</div>
```

AngularJS numbers are like JavaScript numbers:

```
<div ng-app="" ng-init="quantity=1;cost=5">
<p>Total in dollar: {{ quantity * cost }}</p>
</div>
```

Same example using ng-bind:

```
<div ng-app="" ng-init="quantity=1;cost=5">
<p>Total in dollar: <span ng-bind="quantity * cost"></span></p>
</div>
```

AngularJS Expressions vs. JavaScript Expressions

Unlike JavaScript expressions, AngularJS expressions can be written inside HTML.

AngularJS expressions do not support conditionals, loops, and exceptions, while JavaScript expressions do.

AngularJS expressions support filters, while JavaScript expressions do not.

AngularJS Modules

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

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

<script>

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

</script>
```

The "myApp" parameter refers to an HTML element in which the application will run.

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

```
<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>
```

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:

```
<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>
```

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:

```
<!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>
```

myApp.js

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

The [] parameter in the module definition can be used to define dependent modules.

Without the [] parameter, you are not creating a new module, but retrieving an existing one.

myCtrl.js

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

  $scope.firstName = "John";

  $scope.lastName= "Doe";

});
```

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

Data Binding

Data binding in AngularJS binds AngularJS expressions with AngularJS data.

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

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-app directive defines the **root element** of an AngularJS application.

The ng-app directive will **auto-bootstrap** (automatically initialize) the application when a web page is loaded.

Create New Directives

New directives are created by using the .directive function. To invoke the new directive, make an HTML element with the same tag name as the new directive. When naming a directive, you must use a camel case name, **w3TestDirective**, but when invoking it, you must use **-** separated name, w3-test-directive:

```
<body ng-app="myApp">
<w3-test-directive></w3-test-directive>
<script>
var app = angular.module("myApp", []);
app.directive("w3TestDirective", function() {
 return {
  template : "<h1>Made by a directive!</h1>"
 };
});
</script>
</body>
```

The ng-model directive can provide type validation for application data (number, e-mail, required):

```
<form ng-app="" name="myForm">

  Email:

  <input type="email" name="myAddress" ng-model="text">

  <span ng-show="myForm.myAddress.$error.email">Not a valid e-mail address</span>

</form>
```

The data model is a collection of data available for the application. The HTML container where the AngularJS application is displayed, is called the view. The view has access to the model, and there are several ways of displaying model data in the view.

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.

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.

```
<!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 **orderBy** filter sorts an array:

```
<!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>

<ul>

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

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

  </li>

</ul>


</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>
```

Return the names that contains the letter "i":

```
<!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>
```

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>
```

Use the $http service to request data from the server:

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

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

  $http.get("welcome.htm").then(function (response) {

    $scope.myWelcome = response.data;

  });

});
```

The $timeout Service

The $timeout service is AngularJS' version of the window.setTimeout function. Display a new message after two seconds:

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

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

  $scope.myHeader = "Hello World!";

  $timeout(function () {

    $scope.myHeader = "How are you today?";

  }, 2000);

});
```

Display the time every second:

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

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

  $scope.theTime = new Date().toLocaleTimeString();
  $interval(function () {
    $scope.theTime = new Date().toLocaleTimeString();
  }, 1000);
});
```

Displaying Data in a Table

```
<!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="customersCtrl">

<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('customersCtrl', function($scope, $http) {
    $http.get("customers.php")
  .then(function (response) {$scope.names = response.data.records;});
});
</script>

</body>
</html>
```

<u>To display the table index, add a <td> with $index:</u>

```
<!DOCTYPE html>
<html>
<style>
table, th , td  {
 border: 1px solid grey;
 border-collapse: collapse;
 padding: 5px;
}
table tr:nth-child(odd) {
 background-color: #f1f1f1;
}
table tr:nth-child(even) {
 background-color: #ffffff;
}
</style>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
```

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


<table>
  <tr ng-repeat="x in names">
    <td>{{ $index + 1 }}</td>
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>


</div>


<script>
var app = angular.module('myApp', []);
app.controller('customersCtrl', function($scope, $http) {
    $http.get("customers.php")
    .then(function (response) {$scope.names = response.data.records;});
});
</script>


</body>
</html>
```

Using $even and $odd

```
<table>
```

```
 <tr ng-repeat="x in names">
  <td ng-if="$odd" style="background-color:#f1f1f1">{{ x.Name }}</td>
  <td ng-if="$even">{{ x.Name }}</td>
  <td ng-if="$odd" style="background-color:#f1f1f1">{{ x.Country }}</td>
  <td ng-if="$even">{{ x.Country }}</td>
 </tr>
</table>
```

To create a dropdown list, based on an object or an array in AngularJS, you should use the ng-options directive:

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

<select ng-model="selectedName" ng-options="x for x in names">
</select>

</div>

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

You can also use the ng-repeat directive to make the same dropdown list:

```
<select>
```

```
  <option ng-repeat="x in names">{{x}}</option>

</select>
```

In the previous examples the data source was an array, but we can also use an object. Assume you have an object with key-value pairs:

```
$scope.cars = {

  car01 : "Ford",

  car02 : "Fiat",

  car03 : "Volvo"

};
```

Using an object as the data source, **x** represents the key, and **y** represents the value:

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

</select>

<h1>You selected: {{selectedCar}}</h1>
```

AngularJS is perfect for displaying data from a Database. Just make sure the data is in JSON format.

Fetching Data From a PHP Server Running MySQL

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


<table>

  <tr ng-repeat="x in names">

    <td>{{ x.Name }}</td>

    <td>{{ x.Country }}</td>

  </tr>

</table>


</div>
```

```
<script>

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

app.controller('customersCtrl', function($scope, $http) {

  $http.get("customers_mysql.php")

  .then(function (response) {$scope.names = response.data.records;});

});

</script>
```

Cross-Site HTTP Requests

A request for data from a different server (other than the requesting page), are called cross-site HTTP requests.

Cross-site requests are common on the web. Many pages load CSS, images, and scripts from different servers.

In modern browsers, cross-site HTTP requests from scripts are restricted to same site for security reasons.

The following line, in our PHP examples, has been added to allow cross-site access.

```
header("Access-Control-Allow-Origin: *");
```

Server Code ASP.NET, VB and MS Access

```
<%@ Import Namespace="System.IO"%>

<%@ Import Namespace="System.Data"%>

<%@ Import Namespace="System.Data.OleDb"%>

<%

Response.AppendHeader("Access-Control-Allow-Origin", "*")

Response.AppendHeader("Content-type", "application/json")

Dim conn As OleDbConnection

Dim objAdapter As OleDbDataAdapter
Dim objTable As DataTable

Dim objRow As DataRow

Dim objDataSet As New DataSet()

Dim outp

Dim c

conn = New OledbConnection("Provider=Microsoft.Jet.OLEDB.4.0;data
source=Northwind.mdb")

objAdapter = New OledbDataAdapter("SELECT CompanyName, City, Country FROM
Customers", conn)

objAdapter.Fill(objDataSet, "myTable")

objTable=objDataSet.Tables("myTable")


outp = ""

c = chr(34)

for each x in objTable.Rows

if outp <> "" then outp = outp & ","

outp = outp & "{" & c & "Name"    & c & ":" & c & x("CompanyName") & c & ","

outp = outp &      c & "City"    & c & ":" & c & x("City")       & c & ","

outp = outp &      c & "Country" & c & ":" & c & x("Country")    & c & "}"

next


outp ="{" & c & "records" & c & ":[" & outp & "]}"

response.write(outp)

conn.close

%>
```

<u>Server Code ASP.NET, Razor C# and SQL Lite</u>

```
@{

Response.AppendHeader("Access-Control-Allow-Origin", "*")

Response.AppendHeader("Content-type", "application/json")

var db = Database.Open("Northwind");

var query = db.Query("SELECT CompanyName, City, Country FROM Customers");

var outp =""

var c = chr(34)

}

@foreach(var row in query){

if (outp != "") {outp = outp + ","}

outp = outp + "{" + c + "Name"    + c + ":" + c + @row.CompanyName + c + ","

outp = outp +      c + "City"    + c + ":" + c + @row.City       + c + ","

outp = outp +      c + "Country" + c + ":" + c + @row.Country    + c + "}"

}

outp ="{" + c + "records" + c + ":[" + outp + "]}"

@outp
```

Use the HTML5 attribute required to specify that the input field must be filled out:

```
<!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>
```

Apply styles, using standard CSS:

```
<!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>
```

With AngularJS, you can include HTML content using the **ng-include** directive:

```
<!DOCTYPE html>

<html>

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

<body ng-app="">


<div ng-include="'myFile.htm'"></div>


</body>

</html>
```

To make your applications ready for animations, you must include the AngularJS Animate library:

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

Then you must refer to the ngAnimate module in your application:

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

CSS Transitions

CSS transitions allows you to change CSS property values smoothly, from one value to another, over a given duration.

When the DIV element gets the .ng-hide class, the transition will take 0.5 seconds, and the height will smoothly change from 100px to 0:

```
<!DOCTYPE html>

<html>

<style>

div {

  transition: all linear 0.5s;

  background-color: lightblue;

  height: 100px;

}


.ng-hide {

  height: 0;

}


</style>

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

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


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

```
<h1>Hide the DIV: <input type="checkbox" ng-model="myCheck"></h1>

<div ng-hide="myCheck"></div>

<script>
var app = angular.module('myApp', ['ngAnimate']);
</script>

</body>
</html>
```

Style your application using the W3.CSS stylesheet:

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

AngularJS Routing

The **ngRoute** module helps your application to become a Single Page Application. If you want to navigate to different pages in your application, but you also want the application to be a SPA (Single Page Application), with no page reloading, you can use the ngRoute module.

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

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

```
<p><a href="#/!">Main</a></p>

<a href="#!red">Red</a>
<a href="#!green">Green</a>
<a href="#!blue">Blue</a>

<div ng-view></div>

<script>
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
   $routeProvider
   .when("/", {
      templateUrl : "main.htm"
   })
   .when("/red", {
      templateUrl : "red.htm"
   })
   .when("/green", {
      templateUrl : "green.htm"
   })
   .when("/blue", {
      templateUrl : "blue.htm"
   });
});
</script>
```

```
<p>Click on the links to navigate to "red.htm", "green.htm", "blue.htm", or back to
"main.htm"</p>

</body>

</html>
```