AngularJS is a **JavaScript framework**. It can be added to an HTML page with a <script> tag. AngularJS extends HTML with **ng-directives(ng-app, ng-model, ng-bind, ng-init, ng-controller)**.

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

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

**AngularJS Extends HTML:**

**Eg:**

<div ng-app="" ng-init="firstName='John'">

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

<h1>Hello {{name}}</h1>

<p>The name is <span ng-bind="firstName"></span>

</div>

* 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 **innerHTML** of the element to the application variable **name**.
* The **ng-init** directive initializes AngularJS application variables.

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

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

## AngularJS Expressions:

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

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

Ex:

<div ng-app="">  
  <p>My first expression: {{ 5 + 5 }}</p>  
</div>

**AngularJS Applications:**

* AngularJS **modules** define AngularJS applications.
* AngularJS **controllers** control AngularJS applications.
* The **ng-app** directive defines the application; the **ng-controller** directive defines the controller.

**AngularJS Example**

<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>

AngularJS modules define applications:

**AngularJS Module**

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

AngularJS controllers control applications:

**AngularJS Controller**

app.controller('myCtrl', function($scope) {  
    $scope.firstName= "John";  
    $scope.lastName= "Doe";  
});

\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\* Day1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## AngularJS Expressions

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

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

AngularJS will "output" data 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 }}

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

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

## AngularJS Numbers

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>

Using **ng-init** is not very common. You will learn a better way to initialize data in the chapter about controllers.

## AngularJS Strings

AngularJS strings are like JavaScript strings:

<div ng-app="" ng-init="firstName='John';lastName='Doe'">  
  
 <p>The name is {{ firstName + " " + lastName }}</p>  
  
</div>

Same example using ng-bind:

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

## AngularJS Objects

AngularJS objects are like JavaScript objects:

<div ng-app="" ng-init="person={firstName:'John', lastName:'Doe'}">  
  
 <p>The name is {{ person.lastName }}</p>  
  
</div>

Same example using ng-bind:

<div ng-app="" ng-init="person={firstName:'John', lastName:'Doe'}">  
  
 <p>The name is <span ng-bind="person.lastName"></span></p>  
  
</div>

## AngularJS Arrays

AngularJS arrays are like JavaScript arrays:

<div ng-app="" ng-init="points=[1,15,19,2,40]">  
  
 <p>The third result is {{ points[2] }}</p>  
  
</div>

Same example using ng-bind:

<div ng-app="" ng-init="points=[1,15,19,2,40]">  
  
 <p>The third result is <span ng-bind="points[2]"></span></p>  
  
</div>

## AngularJS Expressions vs. JavaScript Expressions

Like JavaScript expressions, AngularJS expressions can contain literals, operators, and variables.

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

Unlike JavaScript expressions, AngularJS expressions do not support conditionals, loops, or exceptions.

Unlike JavaScript expressions, AngularJS expressions support filters.

\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\* Day2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# AngularJS Directives

AngularJS lets you extend HTML with new attributes called **Directives**.

AngularJS directives are extended HTML attributes with the prefix **ng-**.

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, text area) to application data.

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

## Data Binding

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

Data binding in AngularJS synchronizes AngularJS expressions with AngularJS data.

**{{ firstName }}** is synchronized with **ng-model="firstName"**.

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

Eg:

<div ng-app="" ng-init="quantity=1;price=5">  
  
Quantity: <input type="number" ng-model="quantity">  
Costs:    <input type="number" ng-model="price">  
  
Total in dollar: {{ quantity \* price }}  
  
</div>

## Repeating HTML Elements:

## 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>

AngularJS is perfect for database CRUD (Create Read Update Delete) applications.  
Just imagine if these objects were records from a database.

## The ng-app Directive:

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.

Later you will learn how **ng-app** can have a value (like ng-app="myModule"), to connect code modules.

## The ng-init Directive:

The **ng-init** directive defines **initial values** for an AngularJS application.

Normally, you will not use ng-init. You will use a controller or module instead.

You will learn more about controllers and modules later.

## The 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.

## The ng-repeat Directive

The **ng-repeat** directive **clones HTML elements** once for each item in a collection (in an array).

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Directives Done \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 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**.

<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>

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):

<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>

## 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](http://www.w3schools.com/angular/personController.js):

## <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: {{firstName + " " + lastName}} </div> <script src="personController.js"></script>

## In personController.js put the following code:

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

## Iterating through array of Objects in external java script file and calling in our controller.

## <div ng-app="myApp" ng-controller="namesCtrl"> <ul>   <li ng-repeat="x in names">     {{ x.name + ', ' + x.country }}   </li> </ul> </div> <script src="namesController.js"></script>

## In [namesController.js](http://www.w3schools.com/angular/namesController.js) put the following code:

## angular.module('myApp', []).controller('namesCtrl', function($scope) {     $scope.names = [         {name:'Jani',country:'Norway'},         {name:'Hege',country:'Sweden'},         {name:'Kai',country:'Denmark'}     ]; });

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Controllers Done \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# AngularJS Filters:

Filters can be added to expressions and directives using a pipe character.

AngularJS Filters

AngularJS filters can be used to transform data:

|  |  |
| --- | --- |
| **Filter** | **Description** |
| currency | Format a number to a currency format. |
| Filter | Select a subset of items from an array. |
| lowercase | Format a string to lower case. |
| orderBy | Orders an array by an expression. |
| uppercase | Format a string to upper case. |

## Adding Filters to Expressions:

A filter can be added to an expression with a pipe character (|) and a filter.

(For the next two examples we will use the person controller from the previous chapter)

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

<div ng-app="myApp" ng-controller="personCtrl">  
<p>The name is {{ lastName | uppercase }}</p>

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

## The currency Filter

The **currency** filter formats a number as currency:

## <div ng-app="myApp" ng-controller="costCtrl">

## <input type="number" ng-model="quantity">

## <input type="number" ng-model="price">

## <p>Total = {{ (quantity \* price) | currency }}</p>

## <p>Total = {{ total() }}</p>

## </div>

## <script>

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

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

## $scope.quantity = 2;

## $scope.price = 4;

## $scope.total = function() {

## return $scope.price \* $scope.quantity;

## }

## });

## </script>

## Adding Filters to Directives:

A filter can be added to a directive with a pipe character (|) and a filter.

The **orderBy** filter orders an array by an expression:

## <div ng-app="myApp" ng-controller="namesCtrl"> <ul>   <li ng-repeat="x in names | orderBy:'country'">     {{ x.name + ', ' + x.country }}   </li> </ul> <div>

## <script>

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

## app.controller('namesCtrl', function($scope) {

## $scope.names = [

## {name:'Ashish', country:'India'},

## {name:'Jose', country:'Brazil'},

## {name:'Leone', country:'Canada'},

## {name:'Waugh', country:'Australia'}

## ]

## });

## </script>

## Filtering Input:

An input filter can be added to a directive with a pipe character (|) and filter followed by a colon and a model name.

The **filter** filter selects a subset of an array:

<div ng-app="myApp" ng-controller="namesCtrl">  
  
<p><input type="text" ng-model="search"></p>  
  
<ul>  
  <li ng-repeat="x in names | filter:search | orderBy:'country'">  
    {{ (x.name | uppercase) + ', ' + x.country }}  
  </li>  
</ul>  
  
</div>

<script>

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

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

$scope.names = [

{name:'Ashish', country:'India'},

{name:'Jose', country:'Brazil'},

{name:'Leone', country:'Canada'},

{name:'Waugh', country:'Australia'}

]

});

</script>

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Filter Over \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# AngularJS AJAX - $http:

## ****$http**** is an AngularJS service for reading data from remote servers.

## Providing Data

The following data can be provided by a web server:

## http://www.w3schools.com/angular/customers.php

## AngularJS $http

AngularJS **$http** is a core service for reading data from web servers.

$http.get(url) is the function to use for reading server data.

## $http is an **XMLHttpRequest object** for requesting external data.

## Eg:

## <div ng-app="myApp" ng-controller="customersCtrl">  <ul>   <li ng-repeat="x in names">     {{ x.Name + ', ' + x.Country }}   </li> </ul> </div> <script> var app = angular.module('myApp', []); app.controller('customersCtrl', function($scope, $http) {     $http.get("http://www.w3schools.com/angular/customers.php")     .success(function(response) {$scope.names = response.records;}); }); </script>

Application explained:

* The AngularJS application is defined by **ng-app**. The application runs inside a <div>.
* The **ng-controller** directive names the **controller object**.
* The **customersCtrl** function is a standard JavaScript **object constructor**.
* AngularJS will invoke customersCtrl with a **$scope** and **$http** object.
* $scope is the **application object** (the owner of application variables and functions).
* $http is an **XMLHttpRequest object** for requesting external data.
* **$http.get()** reads **JSON data** from <http://www.w3schools.com/angular/customers.php>.
* If **success**, the controller creates a property (**names**) in the scope, with JSON data from the server.

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* $http over \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# AngularJS Tables:

The ng-repeat directive is perfect for displaying tables.

Displaying Data in a Table

Displaying tables with angular is very simple:

## <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) {     $http.get("http://www.w3schools.com/angular/customers.php")     .success(function (response) {$scope.names = response.records;}); }); </script>

## Displaying with CSS Style

To make it nice, add some CSS to the page:

## <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>

## Display with orderBy Filter

To sort the table, add an **orderBy** filter:

## <table>   <tr ng-repeat="x in names | orderBy : 'Country'">     <td>{{ x.Name }}</td>     <td>{{ x.Country }}</td>   </tr> </table>

## Display with uppercase Filter

To display uppercase, add an **uppercase** filter:

## Display the Table Index ($index)

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

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

## Using $even and $odd:

## Display odd rows with color and even rows plane.

## <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>

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Tables Over \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# AngularJS HTML DOM:

AngularJS has directives for binding application data to the attributes of HTML DOM elements.

## The ng-disabled Directive

The **ng-disabled** directive binds AngularJS application data to the disabled attribute of HTML elements.

<div ng-app="">  
<p>  
<button ng-disabled="mySwitch">Click Me!</button>  
</p>  
<p>  
<input type="checkbox" ng-model="mySwitch">Button  
</p>  
</div>

Application explained:

The **ng-disabled** directive binds the application data **mySwitch** to the HTML button's **disabled** attribute.

The **ng-model** directive binds the value of the HTML checkbox element to the value of **mySwitch**.

If the value of **mySwitch** evaluates to **true**, the button will be disabled:

## <p> <button disabled>Click Me!</button> </p>

## If the value of **mySwitch** evaluates to **false**, the button will not be disabled:

## <p> <button>Click Me!</button> </p>

## The ng-show Directive:

The **ng-show** directive shows or hides an HTML element.

<div ng-app="">  
<p ng-show="true">I am visible.</p>  
<p ng-show="false">I am not visible.</p>  
</div>

The ng-show directive shows (or hides) an HTML element based on the **value** of ng-show.

You can use any expression that evaluates to true or false:

<div ng-app="" ng-init="hour=11">

<p ng-show="hour > 12">I am visible.</p>

</div>

## The ng-hide Directive

The **ng-hide** directive hides or shows an HTML element.

## <div ng-app=""> <p ng-hide="true">I am not visible.</p> <p ng-hide="false">I am visible.</p> </div>

# AngularJS Animations

## AngularJS provides animated transitions, with help from CSS.

## An animation is when the transformation of an HTML element gives you an illusion of motion

|  |  |
| --- | --- |
| **Note** | Applications should not be filled with animations, but some animations can make the application easier to understand. |

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

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

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

<body ng-app="ngAnimate">

Or if your application has a name, add ngAnimate as a dependency in your application module:

### **Example**

<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>