AngularJS Directives

[« Previous](http://www.w3schools.com/angular/angular_modules.asp)

[Next Chapter »](http://www.w3schools.com/angular/angular_model.asp)

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

AngularJS has a set of built-in directives which offers functionality to your applications.

AngularJS also lets you define your own directives.

AngularJS 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, textarea) to application data.

Read about all AngularJS directives in our [AngularJS directive reference](http://www.w3schools.com/angular/angular_ref_directives.asp).

Example

<div ng-app="" ng-init="firstName='John'">  
  
<p>Name: <input type="text" ng-model="firstName"></p>  
<p>You wrote: {{ firstName }}</p>  
  
</div>

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_directives)

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 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:

Example

<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>

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_binding)

|  |  |
| --- | --- |
| **Note** | Using ng-init is not very common. You will learn how to initialize data in the chapter about controllers. |

Repeating HTML Elements

The ng-repeat directive repeats an HTML element:

Example

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

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_repeat_array)

The ng-repeat directive actually **clones HTML elements** once for each item in a collection.

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

Example

<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>

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_repeat_object)

|  |  |
| --- | --- |
| **Note** | 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.

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.

Read more about the ng-model directive in the next chapter.

Create New Directives

In addition to all the built-in AngularJS directives, you can create your own 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:

Example

<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>

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_directive_tagname)

You can invoke a directive by using:

* Element name
* Attribute
* Class
* Comment

The examples below will all produce the same result:

Element name

<w3-test-directive></w3-test-directive>

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_directive_element)

Attribute

<div w3-test-directive></div>

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_directive_attribute)

Class

<div class="w3-test-directive"></div>

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_directive_class)

Comment

<!-- directive: w3-test-directive -->

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_directive_comment)

Restrictions

You can restrict your directives to only be invoked by some of the methods.

Example

By adding a restrict property with the value "A", the directive can only be invoked by attributes:

var app = angular.module("myApp", []);  
app.directive("w3TestDirective", function() {  
    return {  
        restrict : "A",  
        template : "<h1>Made by a directive!</h1>"  
    };  
});

[Try it Yourself »](http://www.w3schools.com/angular/tryit.asp?filename=try_ng_directive_restrictions)

The legal restrict values are:

* E for Element name
* A for Attribute
* C for Class
* M for Comment

By default the value is EA, meaning that both Element names and attribute names can invoke the directive.