AngularJS Scope

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

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

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.

The scope is available for both the view and the controller.

How to Use the Scope?

When you make a controller in AngularJS, you pass the $scope object as an argument:

Example

Properties made in the controller, can be referred to in the view:

<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>

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

When adding properties to the $scope object in the controller, the view (HTML) gets access to these properties.

In the view, you do not use the prefix $scope, you just refer to a propertyname, like {{carname}}.

Understanding the Scope

If we consider an AngularJS application to consist of:

* View, which is the HTML.
* Model, which is the data available for the current view.
* Controller, which is the JavaScript function that makes/changes/removes/controls the data.

Then the scope is the Model.

The scope is a JavaScript object with properties and methods, which are available for both the view and the controller.

Example

If you make changes in the view, the model and the controller will be updated:

<div ng-app="myApp" ng-controller="myCtrl">  
  
<input ng-model="name">  
  
<h1>My name is {{name}}</h1>  
  
</div>  
  
<script>

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

</script>

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

Know Your Scope

It is important to know which scope you are dealing with, at any time.

In the two examples above there is only one scope, so knowing your scope is not an issue, but for larger applications there can be sections in the HTML DOM which can only access certain scopes.

Example

When dealing with the ng-repeat directive, each repetition has access to the current repetition object:

<div ng-app="myApp" ng-controller="myCtrl">  
  
<ul>  
    <li ng-repeat="x in names">{{x}}</li>  
</ul>  
  
</div>  
  
<script>

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

</script>

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

Each <li> element has access to the current repetition object, in this case a string, which is referred to by using x.

Root Scope

All applications have a $rootScope which is the scope created on the HTML element that contains the ng-app directive.

The rootScope is available in the entire application.

If a variable has the same name in both the current scope and in the rootScope, the application use the one in the current scope.

Example

A variable named "color" exists in both the controller's scope and in the rootScope:

<body ng-app="myApp">  
  
<p>The rootScope's favorite color:</p>  
<h1>{{color}}</h1>  
  
<div ng-controller="myCtrl">  
    <p>The scope of the controller's favorite color:</p>  
    <h1>{{color}}</h1>  
</div>  
  
<p>The rootScope's favorite color is still:</p>  
<h1>{{color}}</h1>  
  
<script>

var app = angular.module('myApp', []);  
app.run(function($rootScope) {  
    $rootScope.color = 'blue';  
});  
app.controller('myCtrl', function($scope) {  
    $scope.color = "red";  
});

</script>  
</body>

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