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.

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<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>

<p>Notice that controller's color variable does not overwrite the rootScope's color value.</p>

</body>

</html>

**Usinf Order by filter to sort data byclicking on table geading**

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>

<p>Click the table headers to change the sorting order:</p>

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

<table border="1" width="100%">

<tr>

<th ng-click="orderByMe('name')">Name</th>

<th ng-click="orderByMe('country')">Country</th>

</tr>

<tr ng-repeat="x in names | orderBy:myOrderBy">

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

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

</tr>

</table>

</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'}

];

$scope.orderByMe = function(x) {

$scope.myOrderBy = x;

}

});

</script>

</body>

</html>

Services

Https servcies

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>

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

<p>Today's welcome message is:</p>

<h1>{{myWelcome}}</h1>

</div>

<p>The $http service requests a page on the server, and the response is set as the value of the "myWelcome" variable.</p>

<script>

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

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

$http.get("welcome.htm")

.then(function(response) {

$scope.myWelcome = response.data;

});

});

</script>

</body>

</html>

The response from the server is an object with these properties:

* .config the object used to generate the request.
* .data a string, or an object, carrying the response from the server.
* .headers a function to use to get header information.
* .status a number defining the HTTP status.
* .statusText a string defining the HTTP status.

Dropdowns made with ng-options allows the selected value to be an **object**, while dropdowns made from ng-repeat has to be a string.

## Create Your Own Service

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>

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

<p>The hexadecimal value of 255 is:</p>

<h1>{{hex}}</h1>

</div>

<p>A custom service with a method that converts a given number into a hexadecimal number.</p>

<script>

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

app.service('hexafy', function() {

this.myFunc = function (x) {

return x.toString(16);

}

});

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

$scope.hex = hexafy.myFunc(255);

});

</script>

</body>

</html>

## Use a Custom Service Inside a Filter

Once you have created a service, and connected it to your application, you can use the service in any controller, directive, filter, or even inside other services.

To use the service inside a filter, add it as a dependency when defining the filter:

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>

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

<p>Use a filter when displaying the array [255, 251, 200]:</p>

<ul>

<li ng-repeat="x in counts">{{x | myFormat}}</li>

</ul>

<p>This filter uses a service that converts numbers into hexadecimal values.</p>

</div>

<script>

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

app.service('hexafy', function() {

this.myFunc = function (x) {

return x.toString(16);

}

});

app.filter('myFormat',['hexafy', function(hexafy) {

return function(x) {

return hexafy.myFunc(x);

};

}]);

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

$scope.counts = [255, 251, 200];

});

</script>

</body>

</html>

## JSON

The data you get from the response is expected to be in JSON format.

JSON is a great way of transporting data, and it is easy to use within AngularJS, or any other JavaScript.

Example: On the server we have a file that returns a JSON object containing 15 customers, all wrapped in array called records.

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<body>

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

<ul>

<li ng-repeat="x in myData">

{{ x.Name + ', ' + x.Country }}

</li>

</ul>

</div>

<script>

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

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

$http.get("customers.php").then(function (response) {

$scope.myData = response.data.records;

});

});

</script>

</body>

</html>

**Ng options vs ng repeat**

Dropdowns made with ng-options allows the selected value to be an **object**, while dropdowns made from ng-repeat has to be a string.

## ngularJS Events

You can add AngularJS event listeners to your HTML elements by using one or more of these directives:

* ng-blur
* ng-change
* ng-click
* ng-copy
* ng-cut
* ng-dblclick
* ng-focus
* ng-keydown
* ng-keypress
* ng-keyup
* ng-mousedown
* ng-mouseenter
* ng-mouseleave
* ng-mousemove
* ng-mouseover
* ng-mouseup
* ng-paste

Client-side validation cannot alone secure user input. Server side validation is also necessary.

**Form States**

The following classes are added to, or removed from, input fields:

* ng-untouched The field has not been touched yet
* ng-touched The field has been touched
* ng-pristine The field has not been  modified yet
* ng-dirty The field has been modified
* ng-valid The field content is valid
* ng-invalid The field content is not valid
* ng-valid-key One key for each validation. Example: ng-valid-required, useful when there are more than one thing that must be validated
* ng-invalid-key Example: ng-invalid-required

The following classes are added to, or removed from, forms:

* ng-pristine No fields has not been modified yet
* ng-dirty One or more fields has been modified
* ng-valid The form content is valid
* ng-invalid The form content is not valid
* ng-valid-key One key for each validation. Example: ng-valid-required, useful when there are more than one thing that must be validated
* ng-invalid-key Example: ng-invalid-required

## Custom Validation for forms

n HTML, the new directive will be referred to by using the attribute my-directive.

In the JavaScript we start by adding a new directive named myDirective.

Remember, when naming a directive, you must use a camel case name, myDirective, but when invoking it, you must use - separated name, my-directive.

Then, return an object where you specify that we require  ngModel, which is the ngModelController.

Make a linking function which takes some arguments, where the fourth argument, mCtrl, is the ngModelController,

Then specify a function, in this case named myValidation, which takes one argument, this argument is the value of the input element.

Test if the value contains the letter "e", and set the validity of the model controller to either true or false.

At last, mCtrl.$parsers.push(myValidation); will add the myValidation function to an array of other functions, which will be executed every time the input value changes.

Parser Vs formatter

If you are typing something in the text box the parser will change the data in the model that time(u will see upper case character in the view)

While formatter is called when some change in model is done ie you will need to click a button

## include Cross Domains

By default, the ng-include directive does not allow you to include files from other domains.

To include files from another domain, you can add a whitelist of legal files and/or domains in the config function of your application:

**ngAnimate for dependency inside angular.module**

**<https://www.w3schools.com/angular/angular_routing.asp>**

Applications can only have one ng-view directive, and this will be the placeholder for all views provided by the route.