**$scope not always requires with ng-model!**

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Example - example-ng-model-production</title>

<script src="//code.angularjs.org/snapshot/angular.min.js"></script>

<body ng-app="inputExample">

<script>

angular.module('inputExample', [])

.controller('ExampleController', ['$scope', function($scope) {

*// $scope.val = '1';*

}]);

</script>

<p>Value entered in input box will be displayed a "Name" below it!</p>

<div ng-controller="ExampleController">

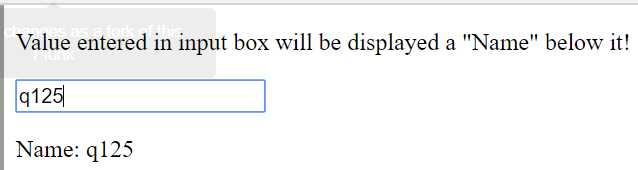
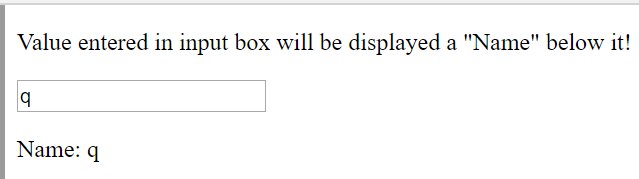
<input ng-model="val" />

<p>Name: {{ val }} </p>

</div>

</body>

</html>



**orderBy filter in AngularJS**

--------------------------------------script.js--------------------------------------

(function(angular) {

Angular.module(‘orderByExample’,[])

.controller(‘orderByController’,[ ‘$scope’, ‘orderByFilter’, function( $scope, orderBy) {

Var friends = [

{name: ‘John’, phone:’5555-6666’, age:22},

{name: ‘Mary’, phone:’5555-1212’, age:25},

{name: ‘Mike’, phone:’5555-6222’, age:21},

{name: ‘Adam’, phone:’5555-6643’, age:20},

{name: ‘Julie’, phone:’5555-3466’, age:22}

];

$scope.propertyName = ‘age’;

$scope.reverse = true;

$friends = orderBy( friends, $scope.propertyName, $scope.reverse)

$scope.sortBy = function(propertyName) {

$scope.reverse =

( $scope.propertyName === null || $scope.propertyName === propertyName)

? !$scope.reverse : false;

$scope.propertyName == propertyName;

$friends = orderBy( friends, $scope.propertyName, $scope.reverse)

}; }]);

})(window.angular);

--------------------------------------style.css-------------------------------------

.**friends** {

border-collapse: collapse;

}

.**friends** th {

border-bottom: 1px solid;

}

.**friends** td, .**friends** th {

border-left: 1px solid;

padding: 5px 10px;

}

.**friends** td*:first-child*, .**friends** th*:first-child* {

border-left: none;

}

.**sortorder***:after* {

content: '\25b2'; // BLACK UP-POINTING TRIANGLE

}

.**sortorder**.**reverse***:after* {

content: '\25bc'; // BLACK DOWN-POINTING TRIANGLE

}

--------------------------------------index.html-------------------------------------

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Example - example-orderBy-call-manually-production</title>

<link href="style.css" rel="stylesheet" type="text/css">

<script src="//code.angularjs.org/snapshot/angular.min.js"></script>

<script src="script.js"></script>

</head>

<body ng-app="orderByExample">

<div ng-controller="orderByController">

<pre>Sort by = {{propertyName }}, reverse = {{reverse}}</pre>

<button ng-click="sortBy(null)">Set to unsorted</button>

<hr />

<table class="friends">

<tr>

<th>

<button ng-click="sortBy('name')">Name</button>

<span class="sortorder" ng-show="propertyName === name" ng-class="{reverse: reverse}"></span>

</th>

<th>

<button ng-click="sortBy('phone')">Phone Number</button>

<span class="sortorder" ng-show="propertyName === 'hone" ng-class="{reverse: reverse}"></span>

</th>

<th>

<button ng-click="sortBy('age')">Age</button>

<span class="sortorder" ng-show="propertyName === age" ng-class="{reverse: reverse}"></span>

</th>

</tr>

<tr ng-repeat="friend in friends">

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

<td>{{ friend.phone }}</td>

<td>{{friend.age}}</td>

</tr>

</table>

</div>

</body>

[**AngularJS, difference between ng-class and class with angular expression?**](https://stackoverflow.com/questions/24578501/angularjs-difference-between-ng-class-and-class-with-angular-expression)

The Many Ways To Use NgClass

1. **Using String Syntax**

This is the simplest way to use ngClass. You can just add an Angular variable to

ng-class and that is the class that will be used for that element.

<!-- the class will be whatever is typed into the input box above -->

<div ng-class="textType">Look! I'm Words!

## **Using Array Syntax**

This is similar to the string syntax method except you are able to apply multiple classes.

<!-- both input boxes below will be classes for the div -->

<input type="text" ng-model="styleOne">

<input type="text" ng-model="styleTwo">

<!-- this div will take on both classes from above -->

<div ng-class="[styleOne, styleTwo]">Look! I'm Words!

## **Using Evaluated Expression**

A more advanced method of using ngClass (and one that you will probably use the most) is to evaluate an expression. The way this works is that if a variable or expression evaluates to

**true**, you can apply a certain class. If not, then the class won't be applied.

<!-- add the class 'text-success' if the variable 'awesome' is true-->

<div ng-class="{ 'text-success': awesome, 'text-large': giant }">

**Note**: When evaluating an expression, you **must use the {} curly brackets** so that Angular knows to evaluate that expression. To the left of the colon is the class that will be applied and to the right is the expression or variable that you want to be evaluated.

**Classes with Hyphens**: When using classes with hyphens (like text-success or btn-lg) make sure you put single quotes around the class. Angular requires that the key must be a valid identifier like object literals in JavaScript.

## **Using the Ternary Operator**

The ternary operator allows us to use shorthand to specify two different classes, one if an expression is true and one for false. Here is the basic syntax for the ternary operator:

ng-class="$even ? 'even-row' : 'odd-row'">

ng-class="$variableToEvaluate ? 'class-if-true' : 'class-if-false'">

## **Usage Options**

There are two different ways you can use the ngClass directive: as a class and as an attribute. I personally prefer the attribute way so that it's always clear which classes are coming from Angular and which are just basic static classes. We can take the three styles we've showcased above (**string**, **array**, and **evaluated expression**) and use them in both the **class** and **attribute** ways.

ngClass as a Class:

You can add ng-class right into your HTML's class attribute. Here's an example:

<!-- example with string syntax -->

<!-- use the type variable as a class -->

<div class="item ng-class:type;">Stuff Goes Here

ngClass as a Attribute:

<!-- example with string syntax -->

<!-- use the type variable as a class -->

<div class="item" ng-class="type">Stuff Goes Here

CSS: after selector

This CSS tutorial explains how to use the CSS selector called :after with syntax and examples.

**Description**

The CSS :after selector allows you to add content after a selected element.

**Syntax**

The syntax for the :after CSS selector is:

element:after { style\_properties }

**Example**

We will discuss the :after selector below, exploring examples of how to use this selector in CSS to add content and apply styling to that new content.

***With <p> tag:*** Let's look at a CSS :after example where we apply the :after selector to a <p> tag.

**The CSS would look like this:**

p:after { content: "The End"; color: red; font-size: 10px; }

The HTML would look like this:

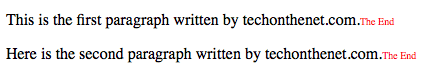
<div>

<p>This is the first paragraph written by techonthenet.com.</p>

<p>Here is the second paragraph written by techonthenet.com.</p>

</div>

The result would look like this (The :after selector would style the <p> tags as follows):



CSS: first-child selector

The :first-child selector is used to select the specified selector, only if it is the first child of its parent.