

### Filter Usage

#### In HTML

```
{{ value | filter_name : arg1 : arg2 }}
```

#### In Javascript

```
$filter('filter_name')(value, arg1, arg2)
```

### currency

**amount**, *number*

Input to filter

**symbol** (?), *string*

Currency symbol or identifier to be displayed

### number

**number**, *number* / *string*

Number to format

**fractionSize** (?), *number* / *string*

Number of decimal places to round the number to. If this is not provided then the fraction size is computed from the current locale's number formatting pattern. Default: 3.

Number of decimal places to round the number to.

### uppercase

**input**, *string*

Converts string to uppercase.

### lowercase

**input**, *string*

Converts string to lowercase.

### limitTo

**input**, *Array* / *string*

Source array or string to be limited.

**limit**, *string* / *number*

The length of the returned array or string.

Creates a new array or string containing only a specified number of elements.

### date

**date**, *Date* / *number* / *string*

Date to format either as Date object, milliseconds (string or number) or various ISO 8601 datetime string formats.

**format** (?), *string*

Formatting rules. If not specified, mediumDate is used.

Formats date to a string based on the requested format.

### json

**object**, \*

Any JavaScript object (including arrays and primitive types) to filter.

Allows you to convert a JavaScript object into JSON string.

### date: format

**yyyy** 4 digit representation of year (e.g. AD 1 => 0001, AD 2010 => 2010)

**yy** 2 digit representation of year, padded (00-99). (e.g. AD 2001 => 01, AD 2010 => 10)

**y** 1 digit representation of year, e.g. (AD 1 => 1, AD 199 => 199)

**MMMM** Month in year (January-December)

**MMM** Month in year (Jan-Dec)

**MM** Month in year, padded (01-12)

**M** Month in year (1-12)

**dd** Day in month, padded (01-31)

**d** Day in month (1-31)

**EEEE** Day in Week,(Sunday-Saturday)

**EEE** Day in Week, (Sun-Sat)

**HH** Hour in day, padded (00-23)

**H** Hour in day (0-23)

**hh** Hour in AM/PM, padded (01-12)

**h** Hour in AM/PM, (1-12)

### date: format (cont)

**mm** Minute in hour, padded (00-59)

**m** Minute in hour (0-59)

**ss** Second in minute, padded (00-59)

**s** Second in minute (0-59)

**.sss** Millisecond in second, padded (000-999)

**a** AM/PM marker

**Z** 4 digit (+sign) representation of the timezone offset (-1200-+1200)

**ww** ISO-8601 week of year (00-53)

**w** ISO-8601 week of year (0-53)

**medium** equivalent to 'MMM d, y h:mm:ss a' for en\_US locale

**short** equivalent to 'M/d/yy h:mm a' for en\_US locale

**fullDate** equivalent to 'EEEE, MMMM d, y' for en\_US locale

**longDate** equivalent to 'MMMM d, y' for en\_US locale

**mediumDate** equivalent to 'MMM d, y' for en\_US locale

**shortDate** equivalent to 'M/d/yy' for en\_US locale

**mediumTime** equivalent to 'h:mm:ss a' for en\_US locale

**shortTime** equivalent to 'h:mm a' for en\_US locale

### filter

**array**, *Array*

The source array.

**expression**, *string* / *Object* / *function()*

The predicate to be used for selecting items from array.

## filter (cont)

**comparator**, *function(actual, expected) | true | undefined*

Comparator which is used in determining if the expected value (from the filter expression) and actual value (from the object in the array) should be considered a match.

Selects a subset of items from array and returns it as a new array.

**filter: expression**

**Expression** — the predicate to be used for selecting items from *array*.

string

The string is evaluated as an expression and the resulting value is used for substring match against the contents of the *array*. All strings or objects with string properties in array that contain this string will be returned. The predicate can be negated by prefixing the string with `!`.

Object

A pattern object can be used to filter specific properties on objects contained by *array*. A special property name \$ can be used (as in {"\$: "text"}) to accept a match against any property of the object. That's equivalent to the simple substring match with a string as described above.

```
function(value, index)
```

A predicate function can be used to write arbitrary filters. The function is called for each element of *array*. The final result is an array of those elements that the predicate returned true for.

## orderBy

**array**, *Array*

The array to sort.

```
expression, function() | string | Array.  
(function()/string)>
```

A predicate to be used by the comparator to determine the order of elements.

**reverse** (?), *boolean*

reverse the order of the array.

Orders a specified array by the expression predicate. It is ordered alphabetically for strings and numerically for numbers.

**orderBy:** expression

A predicate to be used by the comparator to determine the order of elements.

function

Getter function. The result of this function will be sorted using the <, =, > operator.

string

An Angular expression. The result of this expression is used to compare elements (for example name to sort by a property called name or name.substr(0, 3) to sort by 3 first characters of a property called name). The result of a constant expression is interpreted as a property name to be used in comparisons (for example "special name" to sort object by the value of their special name property). An expression can be optionally prefixed with + or - to control ascending or descending sort order (for example, +name or -name). If no property is provided, (e.g. '+') then the array element itself is used to compare where sorting.

## Array

An array of function or string predicates. The first predicate in the array is used for sorting, but when two items are equivalent, the next predicate is used.

### orderBy: expression (cont)

f the predicate is missing or empty then it defaults to '+'.  
 defaults to '+'.