

# AngularJS Filters

Topics : [AngularJS](#)

Written on [January 09, 2024](#)

In AngularJS, filters are used to format and manipulate data in the view. They allow you to transform the appearance of data before it is displayed to the user. AngularJS provides a set of built-in filters, and you can also create custom filters to suit your specific needs.

## Built-in Filters:

### 1. `{{ expression | filter:options }}`:

- **Description:** Filters are applied to expressions in the view using the pipe (|) symbol. They take the form of `{{ expression | filter:options }}`.
- **Example:**

```
<p>{{ message | uppercase }}</p>
```

This example uses the `uppercase` filter to transform the value of the `message` variable to uppercase.

### 2. Common Built-in Filters:

- **currency:** Formats a number as a currency.
- **date:** Formats a date.
- **filter:** Filters an array based on a predicate.
- **json:** Formats an object as JSON.
- **limitTo:** Limits an array/string to a specified number of elements/characters.
- **lowercase and uppercase:** Converts a string to lowercase or uppercase.
- **number:** Formats a number as text.
- **orderBy:** Orders an array by an expression.

Example:

```
<p>{{ price | currency }}</p>
```

This example uses the `currency` filter to format the value of the `price` variable as currency.

## Custom Filters:

You can also create your own custom filters by registering them with the `filter` function within an

AngularJS module.

### 1. Define a Custom Filter:

```
angular.module('myApp').filter('customFilter', function() {  
  return function(input, arg1, arg2) {  
    // Custom filter logic goes here  
    return result;  
  };  
});
```

### 2. Use the Custom Filter in the View:

```
<p>{{ data | customFilter:arg1:arg2 }}</p>
```

In this example, `data` is the variable to be filtered, and `arg1` and `arg2` are optional arguments that can be passed to the custom filter.

## Chaining Filters:

Filters can be chained together to perform multiple transformations on the data.

```
<p>{{ date | date:'mediumDate' | uppercase }}</p>
```

In this example, the `date` filter is applied first to format the date, and then the `uppercase` filter is applied to convert the result to uppercase.

## Example:

Here's a simple example demonstrating the use of built-in filters and a custom filter:

```
<!DOCTYPE html>  
<html lang="en" ng-app="myApp">  
<head>  
<meta charset="UTF-8">  
<title>AngularJS Filters</title>  
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>  
</head>  
<body ng-controller="MyController">  
<p>{{ message | uppercase }}</p>  
<p>{{ currentDate | date:'fullDate' }}</p>  
<p>{{ amount | currency }}</p>  
<p>{{ data | customFilter:'arg1':'arg2' }}</p>  
  
<script>  
angular.module('myApp', []).controller('MyController', function($scope) {  
  $scope.message = 'Hello, Angular!';  
  $scope.currentDate = new Date();  
  $scope.amount = 123.45;  
  $scope.data = 'Some data to be filtered';
```

```
});  
  
angular.module('myApp').filter('customFilter', function() {  
  return function(input, arg1, arg2) {  
    // Custom filter logic (replace spaces with dashes)  
    return input.replace(/\s+/g, '-');  
  };  
});  
</script>  
</body>  
</html>
```

This example uses various built-in filters (uppercase, date, currency) and a custom filter (customFilter) to manipulate and format different types of data in the view.

© Copyright **Aryatechno**. All Rights Reserved. Written tutorials and materials by [Aryatechno](#)