Creating Custom Filters

STEP 1: Define Filter Factory Function

Creates a filtering function

```
function CustomFilterFactory() {
   return function (input) {
     // change input
     return changedInput;
   };
}
```

Return same or changed input

STEP 2: Register Filter Factory With Module

```
angular.module('app', [])
.controller('ctrl', Ctrl)
.filter('custom', CustomFilterFactory);
```

Must be a valid angular expression identifier

STEP 3: (Javascript) Inject it with *name*Filter

```
Ctrl.$inject =
  ['$scope', 'customFilter'];

function Ctrl($scope,
              customFilter) {
  var msg = "Some input";
  customFilter(msg);
};
```

STEP 3: (Javascript) Inject it with *name***Filter**

```
Ctrl.$inject =
   ['$scope', 'customFilter'];

function Ctrl($scope,
              customFilter) {
  var msg = "Some input";
  customFilter(msg);
};
```

STEP 1: Define Filter Factory Function

Creates a filter function

```
function CustomFilterFactory() {
   return function (input) {
     // change input
     return changedInput;
   };
}
```

Filters in Javascript

```
var output =
      $filter('uppercase')(value);
```

Creates filtering function

STEP 2: Register Filter Factory With Module

```
angular.module('app', [])
.controller('ctrl', Ctrl)
.filter('custom', CustomFilterFactory);
```

STEP 3: (Javascript) Inject it with *name***Filter**

```
Ctrl.$inject =
  ['$scope', 'customFilter'];

function Ctrl($scope,
              customFilter) {
  var msg = "Some input";
  customFilter(msg);
};
```

STEP 1: Define Filter Factory Function

Creates a filtering function

```
function CustomFilterFactory() {
  return function (input) {
    // change input
    return changedInput;
  };
}
```

STEP 2: Register Filter Factory With Module

```
angular.module('app', [])
.controller('ctrl', Ctrl)
.filter('custom', CustomFilterFactory);
```

STEP 3: (Javascript) Inject it with *name*Filter

```
Ctrl.$inject =
   ['$scope', 'customFilter'];

function Ctrl($scope,
              customFilter) {
  var msg = "Some input";
  customFilter(msg);
};
```

Creating Custom Filters *With Custom Arguments*

STEP 1: Define Filter (Factory) Function **With Custom Arguments**

```
function CustomFilterFactory() {
  return function (input, arg1) {
           // change input
             return changedInput;
         };
}
```

STEP 2: Register Filter (Factory) Function **With Custom Arguments**

```
angular.module('app', [])
.controller('ctrl', Ctrl)
.filter('custom', CustomFilterFactory);
```

NO CHANGE

STEP 3: (Javascript) Inject it with *name***Filter**

```
Ctrl.$inject =
   ['$scope', 'customFilter'];

function Ctrl($scope,
              customFilter) {
   var msg = "Some input";
   customFilter(msg, "some val");
};
```

STEP 3: (HTML) Use it as registered name

Name the filter **factory** was registered with

```
{{ "Hello" | custom }}
```

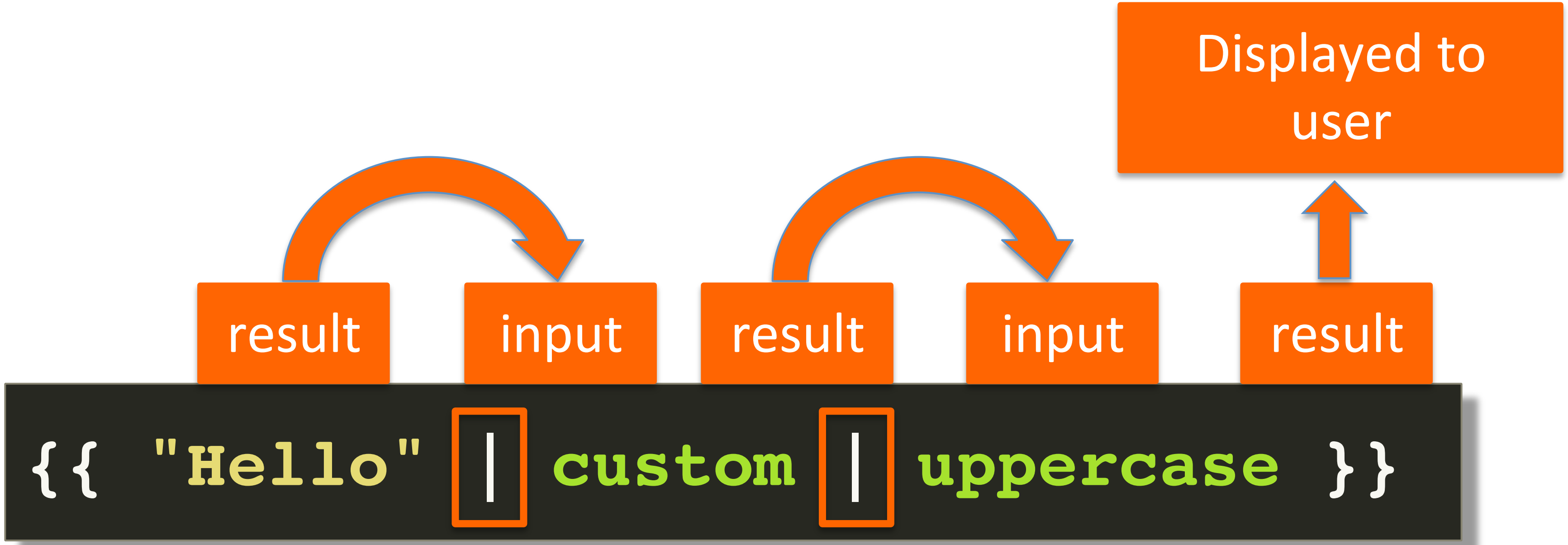
No need to inject filter into the controller

Filters in HTML: Pass extra args with :arg

```
{{ "Hello" | custom: arg1 : arg2 }}
```

Filter specific custom argument

Chaining Filters in HTML



```
{{ "Hello" | custom | uppercase }}
```

Summary

- Steps to create a custom filter
  - Define filter factory function
  - Register filter factory function with module
- To use custom filter in Javascript
  - Inject filter function registeredName*Filter* into controller
- To use in HTML – no need to inject into controller
  - {{ expression | registeredName }}

Summary

- Extra arguments can be supplied to the filter function
  - Otherwise, steps are the same for registration & injection
- To use in HTML with extra arguments
  - {{ expression | registeredName : arg1 : arg2 }}
- Filters can chained!
  - {{ expression | filterOne | filterTwo …}}