In this step you will learn how to create your own custom display filter.

- In the previous step, the details page displayed either "true" or "false" to indicate whether certain phone features were present or not. We have used a custom filter to convert those text strings into glyphs: ✓ for "true", and ✘ for "false". Let's see what the filter code looks like.

Workspace Reset Instructions ➤

The most important changes are listed below. You can see the full diff on GitHub

Custom Filter

In order to create a new filter, you are going to create a `phonecatFilters` module and register your custom filter with this module:

app/js/filters.js :

```
angular.module('phonecatFilters', []).filter('checkmark', function() {
return function(input) {
  return input ? '\u2713' : '\u2718';
};
});
```

The name of our filter is "checkmark". The `input` evaluates to either `true` or `false`, and we return one of the two unicode characters we have chosen to represent true (`\u2713` -> ✓) or false (`\u2718` -> ✘).

Now that our filter is ready, we need to register the `phonecatFilters` module as a dependency for our main `phonecatApp` module.

app/js/app.js :

```
...
angular.module('phonecatApp', ['ngRoute','phonecatControllers','phonecatFilters']);
...
```

Template

Since the filter code lives in the `app/js/filters.js` file, we need to include this file in our layout template.

app/index.html :

```
...
<script src="js/controllers.js"></script>
<script src="js/filters.js"></script>
...
```

The syntax for using filters in Angular templates is as follows:

```
{{ expression | filter }}
```

Let's employ the filter in the phone details template:

app/partials/phone-detail.html :

```
...
<dl>
  <dt>Infrared</dt>
  <dd>{{phone.connectivity.infrared | checkmark}}</dd>
  <dt>GPS</dt>
  <dd>{{phone.connectivity.gps | checkmark}}</dd>
</dl>
...
```

Test

Filters, like any other component, should be tested and these tests are very easy to write.

test/unit/filtersSpec.js :

```
describe('filter', function() {

beforeEach(module('phonecatFilters'));

describe('checkmark', function() {

  it('should convert boolean values to unicode checkmark or cross',
      inject(function(checkmarkFilter) {
    expect(checkmarkFilter(true)).toBe('\u2713');
    expect(checkmarkFilter(false)).toBe('\u2718');
  }));
});
});
```

We must call `beforeEach(module('phonecatFilters'))` before any of our filter tests execute. This call loads our `phonecatFilters` module into the injector for this test run.

Note that we call the helper function, `inject(function(checkmarkFilter) { ... })` , to get access to the filter that we want to test. See angular.mock.inject().

Notice that the suffix 'Filter' is appended to your filter name when injected. See the Filter Guide section where this is outlined.

You should now see the following output in the Karma tab:

```
Chrome 22.0: Executed 4 of 4 SUCCESS (0.034 secs / 0.012 secs)
```

Experiments

Let's experiment with some of the built-in Angular filters and add the following bindings to `index.html` :

- `{{ "lower cap string" | uppercase }}`
- `{{ {foo: "bar", baz: 23} | json }}`
- `{{ 1304375948024 | date }}`
- `{{ 1304375948024 | date:"MM/dd/yyyy @ h:mma" }}`

We can also create a model with an input element, and combine it with a filtered binding. Add the following to index.html:

```
<input ng-model="userInput"> Uppercased: {{ userInput | uppercase }}
```

Summary

Now that you have learned how to write and test a custom filter, go to step 10 to learn how we can use Angular to enhance the phone details page further.

Previous | Live Demo | Code Diff | Next