AngularJS Filters Lab

Formatting data

You are displaying some prices on your site. If you haven't already formatted them, they look fairly ugly.

1. Open checkout.html and find all prices. Where you're displaying any price, go ahead and run it through the currency filter.
2. Since we have to price it in US dollars, add a prefix of "US$" as a parameter. If we don't, the currency prefix will be localized for our users. Yikes!
3. Run and test. Once working, you can move on.

Creating a catalog of products

Let's create a way for our wonderful users to browse through all of our products.

4. If you look in the app/product directory you'll see a file called productList.html. Open that file in your editor.
5. Convert the hardcoded category dropdown to an Angular select list. You can do that kind of like this:

```
<select ng-model='category' name="category" id="category"
        ng-options="category.categoryName for category in categories">
  <option value="">All categories</option>
</select>
```

6. Now clearly we'll need to populate those categories in the controller. So create a controller called productListController.js. Make it a controller in the *productModule* module using all the best practices we've learned so far.
7. Add some categories like this:

```
$scope.categories = [
{"categoryID" : 2, "categoryName" : "Condiments",
  "description" : "Sauces, relishes, spreads, and seasonings"},
{"categoryID" : 4, "categoryName" : "Dairy Products",
  "description" : "Cheeses"},
{"categoryID" : 1, "categoryName" : "Beverages",
  "description" : "Soft drinks, coffees, teas, beers, and ales"},
... and so on
]
```

   Note: look for a file under assets called categories.json.js with some categories. If you copy from it you may save yourself some typing.
8. Make sure everything is wired up properly and test out your view. Can you see your list of categories? Good!
9. In the controller, create a bunch (10 or more) products in an array. Feel free to copy and paste from mainProductController,js.
10. Open your view again and display your $scope's products using ng-repeat. (Note that there are already hardcoded products. Convert that.)
11. Run and test, making sure you can see all 10 items.

Limiting by category

Here comes the payoff. Our goal is that when the user selects a category, he/she only sees the products in that category. To do that we'll employ a filter.

12. Add a "filter" filter to the ng-repeat expression. It should limit the results to only those products whose categoryID is the same as the one the user chooses. It should look kind of like this:

```
| filter: {categoryID: category.categoryID}:true
```

13. The *categoryID* says to match each product's categoryID. the *category* is the one chosen from the dropdown. The *true* means it should be an exact match.
14. Run and test again. Adjust your filter until you get it working.

Adding search ability

We also want our users to be able to search for products on our site. Let's give them a search page. We'll start with the controller.

15. Open the app/product directory and create a new controller file called productSearchController.js.
16. As before create an array of products. Go ahead and copy/paste.
17. You should see a productSearch.html page in that same app/product directory. Go ahead and open that in your editor.
18. Add links to your controller file and whatever else you'll need.
19. Find the input box type="search". Its ng-model should be "searchString". No need for a submit button.
20. Below that, display your $scope's products using ng-repeat.
21. Run and test, making sure you can see all 10 items.

Making the search work

We have a page with 10 products, but the search box does nothing. We can make it work with just a little Angular magic.

22. Add a "filter" filter to the ng-repeat expression. You want to limit the results to only those items that match the search string entered in your search box.
23. Run and test again.

Once you can type into the search box and see your limited list in real time, you can be finished.