AngularJS Capsules Lab

We know that any activities which we may be needed on multiple pages should probably be made a capsule and put under a shared folder. We did that with some services earlier. In this lab we'll get practice with making a factory.

Creating a cart factory

1. Create a new file under *shared* called cartFactory.js. In it create a factory that is part of the shared module. Remember that a factory returns a JSON object. Make sure yours does so.
2. Load that returned JSON object with these methods:
   - getCart()
   - addToCart(product, quantity)
   - removeFromCart(product, quantity)

   These can be method stubs* for now that do nothing more than console.log() that they've been called.

Making the factory work in controllers

They're not doing much yet but let's see how they'd work on a page.

3. Open productDetailController.js. Add a dependency to your new cartFactory.
4. Find where you're handling the AddToCart button click and have it call cartFactory.addToCart(); Don't forget to pass in the product and quantity from the view.
5. Run and test. Make sure it is console.logging the right product and quantity.
6. Open checkoutController.js. Also make it depend on cartFactory.
7. When checkoutController loads you're populating $scope.cart. For now just make a call to cartFactory.getCart().
8. Reload and make sure it also is console.logging.
9. Find the <a> tag where we're removing from the cart. Make its ng-click action make a call to cartFactory.removeFromCart passing the product and quantity. Quantity should be all of that product that is in the cart.
10. Test it out by clicking the trash can icon. Nothing is removed yet but you should see the product and quantity console.logged.
11. Work with these functions until they're all console.logging properly.

See how this is just the same as a service?

---

* A method stub is a function that exists, but doesn't really do anything. Developers use these as placeholders for real functionality later.

Persisting the cart

Well, that was fun, wasn't it? We've created the factory and we're calling factory methods. But the methods don't do anything but console.log. Let's fix that by making use of a RESTful api service.

12. Open cartFactory.js in your editor.
13. In your getCart() method, make an ajax GET call to /api/cart. It'll return a javaScript array of JSON objects with product and quantity. Go ahead and return a promise in getCart().
14. addToCart(product, quantity) should make a POST request to /api/cart. It should pass a JSON object in the body with the product and the quantity.
15. removeFromCart(product, quantity) should make a -- you guessed it -- ajax DELETE call to /api/cart. It should pass a JSON object in the body with the product and the quantity just like with addToCart. Note that since HTTP DELETEs don't normally have a body, you'll have to add this to your request:

```
headers: {"Content-Type": "application/json;charset=utf-8"}
```

16. You'll need to make adjustments to checkoutController.js to implement the callback to getCart(). (Hint: create a .then() with $scope.cart being set to the data). Same with the callback to removeFromCart(). Take care of both of those.
17. Bonus! Do the same with productDetailController.js. When the addToCart() method is run, you may want to produce a message in a .then().