AngularJS Templates Lab

It makes sense to reuse code rather than duplicate it. HTML pages are no exception. We should do that with sections that are common to more than one page. Let's start with the footer.

1. Surf through your site. Notice that the header is exactly the same on every page. And, hey! The footers are, too. Repetition is not good. Let's fix that.
2. Edit your main page, index.html.
3. Go to the bottom of the page and find the footer. Cut it and replace it with this:

```
<footer class="row" ng-include="'/app/shared/pageFooterPartial.html'">
```

4. Now paste it into a new file called pageFooterPartial.html.
5. Run and test. You should still be able to see the page footer.
6. Now reuse this page footer in all other pages (except productDetail.html) on your site by using the ng-include directive.

Replacing the header

You did it with the footer. Now let's do the same with the header.

7. Start with index.html. Find the bounds of the header and copy them into a new file called pageHeaderPartial.html.
8. Use ng-include to include pageHeaderPartial.html. But this time, also give it a controller called pageHeaderController.js.
9. The pageHeaderController doesn't need to do anything except console.log so you can see it running.
10. Do the same for all pages (except productDetail.html) on your site and test them all out. You should see no change except the console.log() messages but now your pages are simpler.

What about productDetail.html?

11. We did this on all pages except productDetail.html because we're not using Angular on that page yet. Let's wire that up.
12. Open productDetail.html in your editor. Using the techniques we've been using since the start, ...
    - Add an ng-app directive pointing to the productModule.
    - Add an ng-controller directive pointing to productDetailController.
    - In the view, change all the hardcoded values to Angular expressions like {{ product.productName }} and {{ product.unitPrice }}
    - Add a filter to that unit price so it looks like a dollar amount.
    - Add an ng-submit directive to the form at the bottom. It should submit to a method called addToCart()
    - Add an ng-model="quantity" directive to the <input> tag.
13. Create productDetailController.js
    - Create productDetailController.js
    - Inject a dependency to productService.

- Hardcode a variable called productID to any number between 1 and 50. Just pick one.
- Make a call to productService.getProduct() so you can get the product data from the RESTful service.
- In the success callback, populate $scope.product.
- Create the addToCart() method. When run, it shold console.log the product and the quantity.

14. Run and test. Make sure you're able to see data from the database for your hardcoded productID. Also when you click the Add to Cart button, your message is logged to the console.

Refactor the headers and footers

Awesome! You completely retrofitted a page and made it work with Angular! Now, finally, you can refactor the header and footer on this page like you did for the others.

15. Replace the header with an ng-include and ng-controller directives.
16. Replace the footer with another ng-include directive.
17. Run and test.

Once all your pages are sharing a common header and footer, you can be finished.