Forms with AngularJS Lab

Add a form to checkout.html

Let's start by modifying the checkout page which has a form for the user's name, address, and a place for a credit card.

1. Open checkout.html.
2. At the bottom of the page, add a with an <h3>Order confirmation</h3>. Add a confirmation message to it that says something like "Your order will be shipped to {{ firstName }} {{ lastName}} at {{ streetAddress }}, ..." Put in there the order total that will be charged to their card.

Modify the checkout controller

3. Open your checkoutController.js file.
4. Add $scope variables for your pretend logged-in customer name, address, city, region/state, zip/postal code, email address.
5. Double-check that processOrder() exists on $scope and that it does a console.log and clears the cart.

Wiring them up

6. Back in the view, connect your <form> fields to the model using ng-model. All the name, address, and email fields should be filled in.
7. Run and test. Can you see the fields being filled in?
8. Make sure your order confirmation at the bottom of the page is populating properly. As you change any fields in the form, that confirmation section should change live. Cool, right?
9. Right now you have the button's click event triggering the order submission but it is cleaner to do that with a form submission. Remove that button click handler and replace it with a form submission event. When the user submits the form, make sure processOrder runs. (Hint: ng-submit).
10. Run and test again. You should see the order details appear in the console and the cart empty.

Adding Angular validations

When you test this page, the HTML form will stop you from submitting if the validations aren't proper. But we should be handling all that from the controller, not the view.

11. Turn off the HTML <form> validation (hint: novalidate).
12. Add this after each required field:

```
<span class="label label-danger">*</span>
```

13. Add an ng-show directive to show each of these spans only if the required field was omitted after the user blurs the field. (Hint: $touched, $invalid, and $error.required).

Bonus!! Setting styles

If you have time, add some CSS to set the background color of an invalid field. You can do this by just setting the background-color to a light red and the color to a red. But if you are using Bootstrap, this is pretty slick.

14. First, note that your form fields are set up like this:

```
<div class="form-group">
  <label for="email" class="control-label">Email</label>
  <input type="email" class="form-control"
         name="email" ng-model="email" required />
</div>
```

15. Then, add something like this to the form-group :

```
ng-class="{ 'has-error': userForm.email.$invalid }"
```

16. Then run and test. Bad data will cause it to turn red. Feel free to optimize it with && and $dirty.

Once you have a well-behaving and validating form, you can be finished. Take a break. You've earned it!