# Forms Validation Lab

## Form validation properties

1. Open checkout.html. Scan through the <input>s for validation properties like required, pattern, type=number, type=email and so forth. Get an idea of what each should do.
2. Note that some of the fields have maxlength="XX". Run and test. Try to put more than the max number of characters in that field. Can you? _____
3. Change them all to ng-maxlength. Try again. Can you now? _____
4. Find every <input> in this page and add this after each one:

```
<span class="glyphicon form-control-feedback">*</span>
```

5. Add an ng-show directive to show each of these spans only if the field is invalid. (Hint: formName.fieldName.$invalid).
6. Run and test. Every one should show the moment an error occurs.
7. It does seem a little rude to show fields in error before the user gets a chance to change them. change your ng-show expression to fix that. (Hint: formName.fieldName.$invalid && formName.fieldName.$touched)
8. Bonus!! If you have time, do the same for the login.html and register.html files.

## More specific help

We're telling the user when a validation problem has happened but we're not telling them what the problem is nor how to solve it. We've got to do something about that.

9. Notice that the shipName input is required and has a max length and has a pattern. Add these below it:

```
<div class="alert alert-danger">Your name is required.</div>
<div class="alert alert-danger">That has a weird character.</div>
<div class="alert alert-danger">Too many characters. Shorten it.</div>
```

10. Now, they will always show but we only want them to show when there is a particular problem. So add a ng-show directive to the first that will only be true when the input has been $touched and formName.fieldName.$error.required is true.
11. Run and test.
12. Do the same for the second field but with a pattern error.
13. And again for the third with a maxlength error.
14. Copy and paste these for each field in your form. Of course, change the message to fit the field and eliminate messages that don't fit the situation.
15. Bonus!! Do the same for the login and register pages.

## Setting styles

Now we're going to add some CSS to draw focus to invalid field. You can do this by just setting. But if you are using Bootstrap, this is pretty slick.

16. First, open site.css and add this:

```
input.ng-invalid.ng-touched { background-color: #FF9494; color: white;}
```

17. Run and test. You will see all fields turn red when in error.
18. Bootstrap gives us some even cooler options. So remove that style from site.css.
19. Then, notice that each field has generally this shape:

```
<div class="form-group has-feedback">
  <label for="shipName" class="control-label">Name to ship to:</label>
  <input ng-model="shipName" name="shipName" class="form-control" />
</div>
```

20. add something like this to the form-group :

```
ng-class="{'has-success': shipTo.shipName.$valid, 'has-error':
shipTo.shipName.$invalid}"
```

21. Then run and test. Bad data will cause it to turn red.
22. Bonus!! Add a $touched requirement to it so it starts out neutral and only turns green or red when the user blurs from the field.
23. Extra bonus!! Add this to all the fields on checkout.html.
24. Super bonus!!! Add this to all the fields in login.html and checkout.html

## Preventing submission

Take a look at the register.html page. As of now, your user could submit the form even when there are validation problems. We can fix that easily enough.

25. Find the submit button on your form. Add something like this to it:

```
ng-disabled="formName.$invalid"
```

26. Run and test.

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