Observables Lab

In this lab we're going to make the ShipOrderComponent work with real data instead of the hardcoded values we've had so far. But before we do, we have one more temporary kludge. We need to set an orderID.

1. Set the ShipOrderComponent as the startup component. (HInt: app.module.ts, set bootstrap to ShipOrderComponent).
2. Create a class scope variable called orderID.
3. In the constructor or NgOnIt, set this.orderID to 11070. Or any orderID you like. Look in the database for some good orderIDs.

Getting locations for those orders

Remember where we're hardcoding a location from which to pick? Let's get an actual location from the database.

4. Open ship-order-component.ts. Find your getBestLocation method where you hardcoded the return.
5. Change that hardcode to hit the GET endpoint for /locations/forProduct/:productID. Of course you're passing the productID in that productID parameter. Make sure you're subscribing a function. In that success callback, set your location.
6. Run and test. When the button is pressed by the user, a real location should be displayed. You'll know you got it right when there's a different location for each product.

Set the shipped or problem flag

7. Still in ship-order-component, find the method you're running when the "Mark as problem" checkbox is checked.
8. Make that send a PATCH request to /api/order/<orderID>/MarkAsProblem
9. Do the same for "Mark as shipped"
10. Run and test them both. When you mark as shipped, you shoud see that the inventory amount is reduced by the amount shipped.

Remember, you can always look at an order and manually set the order status by using mongo directly. Here's a sample for setting the status of order 10 back to 0:

```
db.orders.update( {orderID:10}, {$set: {status:0} } );
```

Receive product

Remember how when we receive product we're displaying the productID and quantity? We'd like to display an entire product on the page instead of merely the productID. Let's go grab a product from our server.

11. As soon as the user enters a productID, make an Ajax GET call to /api/product/<productID>. The response will contain the details for that productID. On success, populate this.product from the response so it will display your product on the form. On failure, show a message that that productID is not found.