Event Binding Lab

When our customers order a product we have to pick the products, pack them, and ship them. Our pickers have to know where to get the product from our warehouse. Let's create a method for them to get the best location for that product.

The ship order component

1. Edit your ship-order.component.ts file. Add a new method called getBestLocation(orderLine). Note that it will receive an orderLine object. For now it should just set orderLine.locationID to "01A1A" and then console.log it.
2. Now edit your ship-order.component.html and find the 'Get best location' button. Add an angular click event handler to it. When fired, it should call the "getBestLocation()" method you just wrote. Don't forget to pass in the current orderLine.
3. Run and test. When you click your button, the button should be replaced by a location.

Cool! Do you see the "Mark as shipped" and "Problem" buttons? Let's work on them.

4. Wire up the "Mark as shipped" button to run a method called markAsShipped(order) which receives in an order object. This method should change the order's status from 0 to 1.
5. Wire up the "Problem" button. It should run a markWithProblem() method which also receives an order object. It should set the order's status to 2.
6. Run and test. You'll know it's working when the Order status and the instructions change as you hit the buttons.

Receive inventory

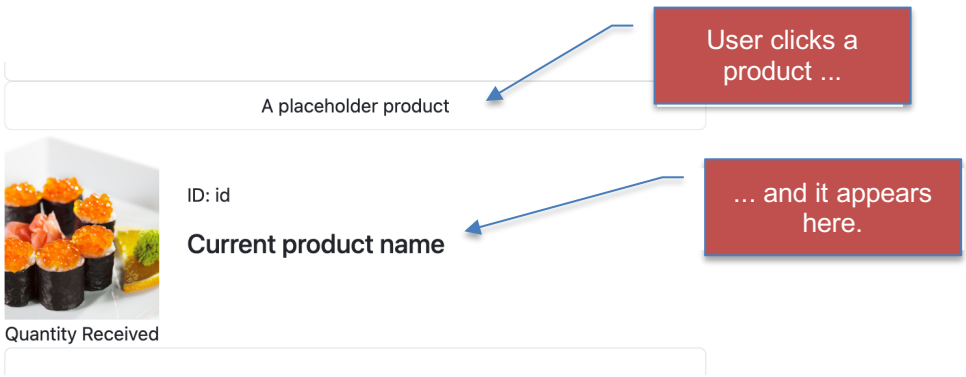
Let's turn our attention to the ReceiveProduct component. This is where the user receives a shipment of product into inventory.

The managers of the warehouse have decided that the users should enter the package tracking number before entering any product. Let's hide everything else on the page until the tracking number is entered.

7. In receive-product.component.ts, create a boolean property called showForm and a string property called trackingNumber.
8. In the template, add a structural directive (hint: *ngIf) so that everything below the tracking number section is gone until this showForm property is true.
9. Create an event handler for the button click. It should call a new method named saveTrackingNumber() which merely sets showForm to true. (Later on we can actually save the tracking number.)
10. Run and test. Did the rest of the component show when you click the button?

Setting the received product

On receive-product.component.html there's a button labelled "A placeholder product". This will eventually be one of many buttons, with each button representing a product. When the user clicks on one of those buttons, they will be "selecting" that to be the currentProduct. But until they select a product from the list, there should be no currentProduct. Let's work on that.



11. First, add a currentProduct property to the class.

```
currentProduct?: Product;
```

12. Next, conditionally show the section if a current product is truthy. (Hint: *ngIf="currentProduct")
13. Make the button's click event call a setCurrentProduct method:

```
setCurrentProduct(product: Product = { id: 0, name: "Fake Product" }) {
  this.currentProduct = product;
}
```

14. Run and test. The section is gone until you click the button.
15. In the currentProduct <section> use interpolation (hint: double curly braces) to display your fake currentProduct's image, id, and name.
16. Run and test one more time to make sure you're seeing your fake product details.

Two final buttons

17. Notice that there's a button labeled "Receive product". When the user clicks it, call a method named receiveProduct(). For now it can just console.log(). We'll make it do something meaningful later.
18. Run and test. Make sure that when you click the button something is logged to the console.
19. Do the same for the "Finished receiving" button. It should call a finishedReceiving() method which just logs something to the console.
20. Run and test that as well.

When you have your buttons working, you can be finished. See how simple event handling can be?