Built-in Directives Lab

Alright, this is where the real fun with Angular begins! In this lab, we're going to iterate through arrays of orders, conditionally display messages and buttons, and conditionally set styles. Let's start with Ship Order Component.

Looping with a *ngFor

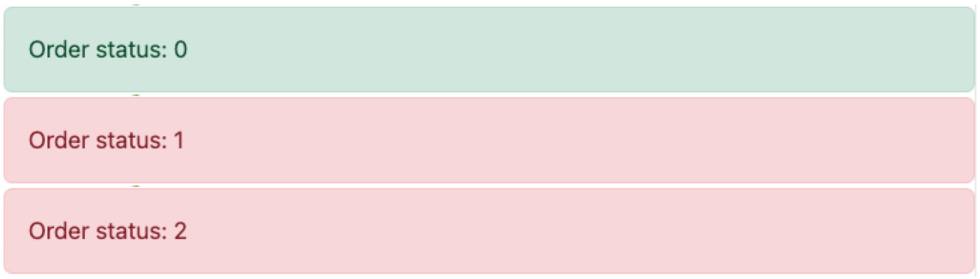
1. Edit your ship-order.component.html. Find where you currently have a hardcoded <tbody>
2. Delete all but one of the <tr>s in that <tbody>. On that remaining <tr>, add a *ngFor directive to loop through the order.lines collection that you created in a previous lab.
3. If you run and test right now, you'll see the same hardcoded line repeated. Give that a try.
4. Now go back in and change the hardcoded values to interpolated values. (Hint: use mustaches). You should see a different productid and picture on each line.

Setting conditional classes with ngClass

There are CSS classes that have been created for you in the Bootstrap CSS library. alert-success is for success messages. alert-danger is for error messages. Let's use them.

5. Take a look at the <div> near the top of ship-order component that shows the order status.

That <div> should have a class of alert-success if the order is ready to pick (order.status is 0) and a class of alert-danger if not. Like this.



Order status: 0

Order status: 1

Order status: 2

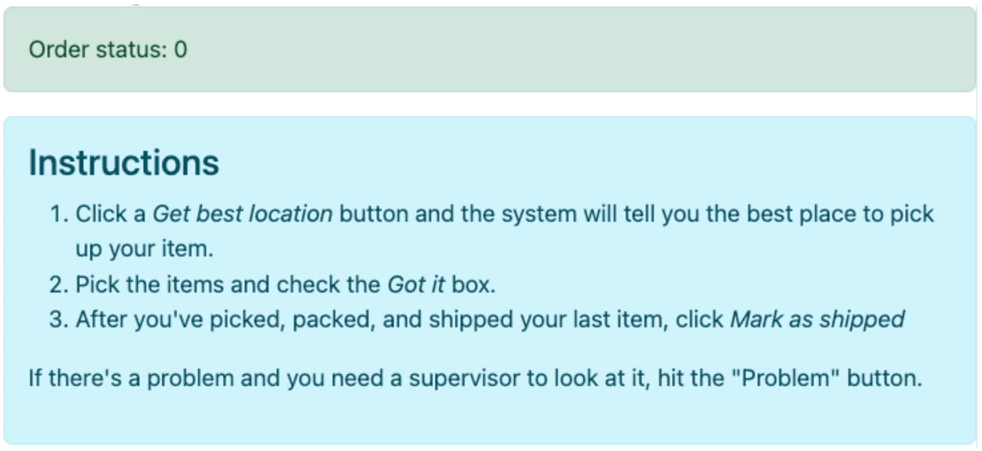
6. Change the template to change the CSS class based on the order status value. (Hint: [ngClass] takes and object in double-quotes. That object's keys are a class name like 'alert', 'alert-success', and 'alert-danger'.)

Using *ngSwitchCase

7. In ship-order.component.html find the instructions. They're in a <section>.

The existing message makes sense if the order status is "0" which means ready to pick. But not if it is "1" which means it has already been shipped or "2" which means that there is a problem with the order. We want the message to change with the status.

For status 0:

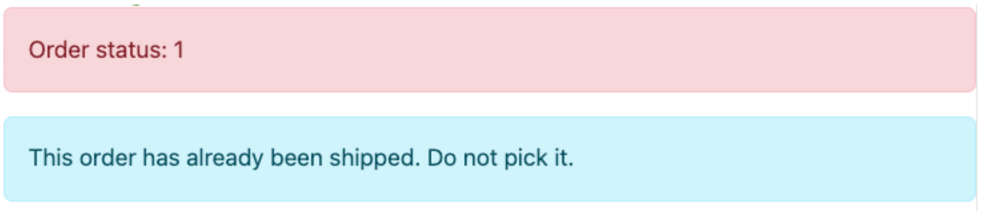

Order status: 0

Instructions

1. Click a *Get best location* button and the system will tell you the best place to pick up your item.
2. Pick the items and check the *Got it* box.
3. After you've picked, packed, and shipped your last item, click *Mark as shipped*

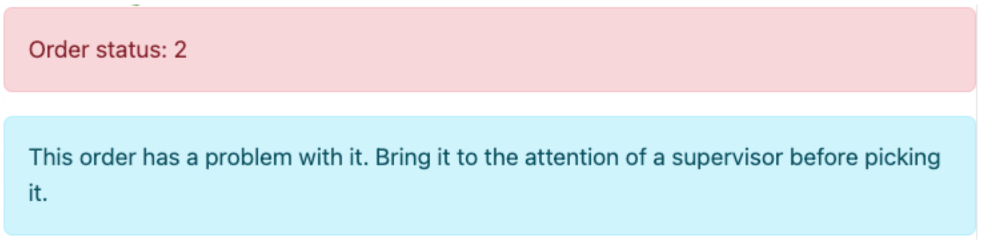
If there's a problem and you need a supervisor to look at it, hit the "Problem" button.

For status 1:


Order status: 1

This order has already been shipped. Do not pick it.

For status 2:


Order status: 2

This order has a problem with it. Bring it to the attention of a supervisor before picking it.

8. Each of the three messages should be wrapped in a <div>. Add all three in individual <div>s inside the <section>.
9. Put an [ngSwitch]="order.status" on the section itself.
10. For the first <div>, put an *ngSwitchCase structural directive to say only show it if the order.status is 0.
11. For the second <div>, put an *ngSwitchCase to say only show it if the order.status is 1.
12. Do the same for the third <div> for when order.status is 2.
13. Run and test, changing the order.status in your TS class. When you're seeing the message change, let's work with those checkboxes.

Use a *ngIf with an else

Eventually our pick/pack/ship process will work like this: The picker sees the order and lines but the location isn't populated. Once they're ready to pick a line/product, they click a button to find the best location for that product and the system populates the locationID in real time.

So if the location exists on the line, we want the locationID to be seen but if not we show a button. Let's do that next.

14. Still in ship-order.component.html, find where you are currently showing a button that says "Get best location".

15. Move that button to a template:

```
<ng-template #getLocation>
  <button class="btn">Get best location</button>
</ng-template>
```

16. Inside the <td>, add a <span> tag with the {{ line.locationID }} in it. Add a *ngIf directive to your <span> tag to say when line.locationID is truthy, show it.
17. Run and test with one or more locationIDs nonexistent. You should see them when they're there and when not, that table cell should be empty.
18. Now add an else clause to your *ngIf. Show the button if line.locationID is falsy.
19. Run and test again. You should see a locationID for the lines that have it and a button for those that don't.

Alright, let's turn our attention to a different component.

Getting a list of orders to display

In the TypeScript lab we made a bunch of orders in the DashBoardComponent. Let's display them on the page.

20. Make DashboardComponent your starting component.
21. Open dashboard.component.html and find the <div>s with the class of order-row. Notice that each of them holds one order. Delete all but one of those and add a *ngFor directive to the one remaining.
22. Replace each value in the <div>s with mustaches/interpolation of the actual values from the order object in the component class.
    - Order column: {{ order.id }}
    - Date column: {{ order.orderDate }}
    - Products column: {{ order.lines.count }}
    - Total column: {{ getOrderTotal(order.lines) }}
23. See that getOrderTotal() method? Add that to dashboard.component.ts:

```
getOrderTotal(lines: Array<OrderLine>) {
  return lines.reduce(
    (p: number, ol: OrderLine) =>
      (p + (ol.price ?? 0) * (ol.quantity ?? 0)),
    0)
}
```

24. Run and test. You should be seeing a list of all the orders on the page with line totals.