

# Built-in Directives Lab

## Looping with a \*ngFor

1. Open ship-order.component.html and 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.line 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). Make sure they look proper to you before you move on.

## 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 a button if not. Let's do that next.

5. Still in ship-order.component.html, find where you are currently showing the line.locationID.
6. Add a \*ngIf directive to your <td> tag to say when locationID is truthy, show it.
7. Run and test with one or more locationIDs nonexistent. You should see them when they're there and when not, your table should be goofy.
8. Now add an ng-template inside the \*ngFor loop. When locationID doesn't exist, show this button:

```
<button class="btn">Get best location</button>
```

9. Run and test again. You should see a locationID for the lines that have it and a button for those that don't.

## Setting a conditional style

10. Make sure there's a <div> near the top of the ship-order component that shows the order status.
11. That 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. Change the template to change the class based on the order status value.
12. There's also a set of instructions. It should have a class of alert-danger if the order.status is 2 (problem status).
13. Run and test with different order statuses.

## Getting a list of orders to display

14. Let's take on a big one! If you look in the setup/assets/codeSnippets folder, you'll see a file called SomeOrders.js. Open the file and copy its contents to the ngOnInit() method of the DashboardComponent.
15. This will populate a property called "orders" in that file. Do a console.log() in ngOnInit() to make sure it is reading alright.

At this point you have a bunch of orders in the DashBoardComponent in a property called orders. Let's display them on the page.

16. Open dashboard.component.html and find the <div> with the class of order-list.
17. Notice that the divs with a class of order-row hold one order each. Delete all but one of those and add a \*ngFor directive.
18. Replace each value in the <div>s with mustaches/interpolation of the actual values from the order object in the component class.
19. Run and test. You should be seeing a list of all the orders on the page.