Expressions Lab

This is where React really gets fun! We have the ability to conditionally display things and to loop through arrays, converting each to JSX to display. Let's try these techniques out with our ever-improving Dinner-and-a-Movie site!

Creating tables and seats

1. Edit your PickSeats.js component. Notice that we've made a private array of tables. Let's expand that into some <Tables> and <Seats>.
2. In the JSX, you'll want to create an expression and use the Array.prototype.map() function to convert that array of objects into an array of <Tables>. Here, this should get you started:

```
tables.map(t => <Table table={t} />)
```

3. Run and test. It should still work ... except that maybe not all the <Seat>s are showing because we previously hardcoded the seats. We should be iterating it as well.
4. Edit Table.js. Instead of having your copy/pasted <Seat> elements, map through them -- again using the array map function to create your <Seat> component instances.
5. Run and test again. Make sure it still works and that you have the right amount of Tables and Seats.

Login and Checkout

6. In the Checkout component, you're immediately pushing the user to the Login component. Change that to behave like this; if a prop called authenticated is truthy, show them the component, otherwise push them to Login. (Hint: you'll want to use && or || in an expression in the JSX).
7. Test it out by passing an authenticated prop to the Checkout component.
8. Hey, while we can still see our Checkout component, let's do some work in here. We'd like to do some conditional rendering.
9. Write another temporary data object:

```
const cart = { contents: { seats: [
  {seat_number: 1, price: 10.75},
  {seat_number: 2, price: 10.75},
  {seat_number: 3, price: 10.75},
] } };
```

10. In the JSX, map through the seats array to draw the <tr>s and <td>s for the table.
11. Run and test.
12. Create a getCartSubtotal() function that returns a number. Where you were previously hardcoding the subtotal, create an expression to call getCartSubtotal() and use the value returned.
13. Do the same with getCartTax() and getCartTotal(), which should just add the tax and subtotal and return it.
14. Run and test. Make sure the numbers make sense as you manually change the contents of your hardcoded cart.
15. Change the JSX by adding an expression. If there are no seats in the cart, Make it say "You have no seats and provide a <Link" to the LandingPage.
16. Run and test with and without seats in the array. Make sure it works properly in either case.

FilmsDetails

17. FilmsDetails has a few fake showtimes. Let's make that a little closer to realistic. First, load it up with some fake showings data:

```
const showings = [
```

```
  {id: 110, film_id: 3, theater_id: 3,
    showing_time: "2019-09-11T21:15:00Z"},
  {id: 111, film_id: 3, theater_id: 3,
    showing_time: "2019-09-11T23:15:00Z"},
  {id: 112, film_id: 3, theater_id: 3,
    showing_time: "2019-09-11T01:15:00Z"}
]
```

18. Then, in the JSX create an expression to loop through those showings using the Array map function. Maybe something like this:

```
showings.map(s => (<li><Link to={"/PickSeats/" + s.id}>
 { new Date(s.showing_time).toLocaleTimeString([],
            { hour: "numeric", minute: "2-digit" }) }
 </Link></li>)
```

19. Run and test. See if you can still see your links, if they look good and if they still work, taking you to a different showing_id each time. If not, adjust until they do.