How to run functions in JSX

1. Edit App.js. We've been displaying just one component at a time so far but now please display two:

```
<PickSeats />
<Checkout { ...state } />
```

Do you see that "{ …state }" thing? It's sort of an advanced JavaScript technique called object spread. When you do this, you're effectively saying "Take the state object, break it down into each of its key/value pairs and send them down into the component" Said differently, this technique says to pass the entire state to Checkout through props. This includes the contents of the user's cart.

We did this because we need to get some values into Checkout and while we could do it through state, props seems cleaner.

2. Have Checkout read the props we need like so:

```
export const Checkout = ({ cart, user }) => {
  console.log("Checkout", cart, user);
```

This is another advanced JavaScript technique called destructuring. The console log is just there for you to debug.

3. Go ahead and test it out. As you click on seats in PickSeats at the top, you should see them appear in the console.log()ged cart.

We've got stuff in the cart so let's display it in the Checkout component.

4. Add a new function in Checkout called makeTableRow():

```
function makeTableRow({seat_id, seat_number, price}) {
  return (
    <tr key={seat_id}>
      <td>{seat_number}</td>
      <td>{price.toCurrency()}</td>
      <td>1</td>
      <td>{price.toCurrency()}</td>
    </tr>
  )
}
```

As you can see, it will receive a seat object, destructure it, and return a table row that will fit nicely in the table.

And now let's call it.

5. Back up in Checkout's <tbody>, you want to create one <tr> for each item in the user's cart. So do something like this:

```
{cart.seats.map(seat => makeTableRow(seat))}
```

6. Run and test. Do you see your rows?

You've just run a function (makeTableRows()) that returns JSX. Let's do another couple of them.

Displaying totals in Checkout

7. Still in Checkout.js find where you have TOTAL_QUANTITY_HERE.

8. We want to replace that with the total quantity, which is just the length of the cart.seats array. Go ahead and write a JSX expression to put {cart.seats.length} in that spot.

9. Run and test. That was easy, right?

10. Well, the price total won't be quite as easy because it is a sum of all of the prices in the cart. If you know the JavaScript Array.prototype.reduce method, go ahead and try to write It without any help. But if you don't, here's a way to get that total:

```
<td>{cart.seats.reduce((total, seat) => total + seat.price, 0)}</td>
```

11. Put that JavaScript in an expression and place it where your other placeholder "TOTAL_MONEY_HERE" is located.

12. Run and test.

13. Bonus! That number should be accurate but it does not look great right now. It should be formatted like a currency. We have such a helper that you've already imported called toCurrency(). Try to implement that.

Once you do, you've implemented not one but two functions being called from JSX. Nice!