Composition Lab

Notice that your LandingPage has some distinct sections, one to choose the date and another to choose a film. Let's separate those into two components, ShowingDate and FilmsList.

Extracting ShowingDate

1. Create a new component called ShowingDate.js.
2. Pull out the JSX surrounding the <input type="date" /> that is used to select the date. All of that needs to go into the new component.
3. Replace all of that JSX with the <ShowingDate /> JSX.
4. If you run and test right now, you should see your component but when you change the date, it no longer works. Let's fix that.
5. Pass a new prop to ShowingDate called pickDate. Pass your function to the pickDate prop kind of like this:

```
<ShowingDate pickDate={chooseDate} />
```

6. Inside ShowingDate.js, change your onClick event to call props.pickDate instead.
7. Run and test. Make adjustments until you get it working.

8. Bonus!! The instructions above asked you to call it pickDate just to help you learn how linkages are made. If it is less confusion to you to keep it named the same in both places (chooseDate), feel free to rename it.

Extracting FilmsList

9. Create a new component called FilmsList.js.
10. Again, extract the JSX for your list of films from LandingPage and put it into FilmsList.js.
11. Remember when we read the films.json file into an object called *films*? Now, let's use it. Pass that whole object into FilmsList kind of like this:

```
<FilmsList films={films} />
```

12. Inside FilmsList.js, where you're displaying the title of your first film, change it to:

```
<h1>{props.film[0].title}</h1>
```

13. Do the same for the poster path and tagline. And repeat for all four of your films.
14. Don't forget your select film event. You'll need to pass the onClick functionality into <FilmsList /> from the LandingPage as a prop.
15. Adjust it so that when the click event occurs, you're passing the entire film from the <FilmsList /> component into the LandingPage's chooseFilm() function.
16. Run and test, adjusting until you get the films showing nicely and the click event working again.

Extracting a Film component

Your FilmsList contains the JSX for a series of Films. Each of those looks like they should be a separate component so lets's extract it. Notice that we'll be creating a grandchild component - a child of FilmsList which is itself a child of LandingPage.

17. Create a new component called Film.js.
18. As before, pull the JSX for displaying one film from FilmsList into Film.js.

You're currently displaying several movies. Each of those should now be a <Film /> tag. Go ahead and start that change. But wait! ...

19. They're all using different posters, titles, and taglines. Since those are all unique to each Film, they should probably be passed into the <Film /> tag as properties. In fact, let's pass it in as a complete object kind of like this:

```
<Film film={films[0]} />
```

20. And of course you'll need to display any film data in curly braces. Here's an example:

```
<img src={props.film.poster_path} alt={props.film.title} />
```

21. Run and test. It should be working as before. But now each of our components is easier to maintain and extend.

Extracting Table and Seat components

You're getting good at this by now. Let's try it with fewer instructions on another component. PickSeats has way too many responsibilities.

22. Edit PickSeats.js. Add an object that looks kind of like this to it:

```
const tables=[
  { id:19, theater_id:2, table_number:4, seats:[1, 2, 3, 4] },
  { id:20, theater_id:2, table_number:6, seats:[5, 6] },
  // Copy and paste the above lines and make them look unique.
  // Create about 15 tables each with 1, 2, or 4 seats.
]
```

23. Create a new component called Table.js. Pull the JSX for a Table out of PickSeats and put it into a new Table component.
24. Have PickSeats's JSX include 3-5 tables, each passing a table object as a prop. (Hint: table={tables[0]})
25. Run and test. Make sure you can still see some tables and seats.
26. Then create a new component called Seat.js. Reach into the JSX for the Table and pull out a Seat component. This time, edit <Table> and pass a seat_id into the <Seat> component.
27. Go back through your <Table>s and <Seat>s and make sure your events still work as expected.

Bonus!! Extracting a Cart component

28. If you have extra time, take a look at your Checkout component. It has a cart in it. Well, the concept of a cart anyway. See if you can extract a <Cart> component from that component. You get to design it any way you like, but here is a thought ... The checkout ability needs to stay with a <Checkout> component. But the displaying of the cart contents should happen in a <Cart> component.