Exercise: Adding table booking state

Overview

Previously, you set up the `BookingForm` component and tracked its state. However, to be truly functional, your Little Lemon reserve-a-table web app functionality will need to be able to share state across components.

Scenario

The goal of this exercise is for you to expand the implementation of the component for the Booking page of the Little Lemon website.

As part of the reserve-a-table web app functionality, the page will display the existing booked table times and available slots, using a list component containing several instances of a `BookingSlot` component.

Available booking slots will be shared between the components and updated when the user submits the form. As your progress through the exercise, it may be worth revising the following lesson items in the **Advanced React** course:

- Working with React hooks
- Lifting up state
- Working with complex data in useState
- What is useReducer and how it differs from useState

Instructions

Step 1: Life state up to the Main component

As you added the table booking state to the `BookingForm` component in the previous exercise, in this exercise, you need to lift the state up to the `Main` component. This is the preferred approach in this case, as your app is relatively simple.

- Move the `availableTimes` `useState` hook from the `BookingForm` component into the `Main` component
- Pass down the state and state changing functions from the `Main` component to the `BookingForm` component using props in order to make state work across different components.

Step 2: Update `BookingForm` to display available times based on the selected date

- The next step is to prepare the available times to be updated based on the date the user has selected. To do this, you will change the `availableTimes` state to a reducer.
- In the `Main` component, create a function named `updateTimes` which will handle the state change. This function will change the `availableTimes` based on the selected date. For now, the function can return the same available times regardless of the date.
- Next, create a function called `initializeTimes` which will create the initial state for the `availableTimes`.

- Then, change `availableTimes` to a reducer using the `useReducer` function and provide the two previous functions as parameters.
- Update the `BookingForm` component to dispatch the state change when the date form field is changed.

Tip: Include the newly selected date in the dispatch parameter.

Conclusion

By completing this exercise, you should now have an app for Little Lemon in which state works across several components, moving you another step closer to having a fully-functional booking page.

Although outside the scope of this capstone project, it's worth mentioning that you may favor more robust state management approaches in the case of larger or more complex apps.