# Exercise: Adding unit tests

## Overview

In this exercise, you will write two unit tests that you can use to check that your web app is working correctly. With regards to writing tests and the related syntax, you may find it useful to recap the following lesson items in the **Advanced React** course:

- Why React Testing library
- Writing the first test for your form

## Scenario

Testing code is a key step of the app development process. In this exercise, you'll practice writing and implementing unit tests for components and state using React Testing Library to verify that the individual functionalities within your Little Lemon web app work as expected.

## Instructions

### Step 1: Test for some static text being rendered in the BookingForm component

- Using your mockups, pick a part of the `BookingForm` component that has some static text, such as a heading or label.
- In preparation for coding a test of this static test, review the following starting code for a test, based on an example in the Writing the first test for your form lesson referenced earlier:

```
import { render, screen } from "@testing-library/react";
import BookingForm from './BookingForm';

test('Renders the BookingForm heading', () => {
    render(<BookingForm />);
    const headingElement = screen.getByText("Book Now");
    expect(headingElement).toBeInTheDocument();
})
```

- Keeping in mind the above example, code a test for the static text being rendered in the `BookingForm` component, using code like this:

```
screen.getByText("BookingForm");
```

**Note:** You will need to adjust the code based on what you've decided your Bookings component should render.

### Step 2: Test the updateTimes and initializeTimes functions

The next step is to validate the behavior of the `updateTimes` and `initializeTimes` reducer functions.

- Write a unit test for the `initializeTimes` function to validate that it returns the correct expected value.
- Write a unit test for the `updateTimes` function to validate that it returns the same value that is provided in the

state. This unit test is important as it will be updated later when the logic of changing the available times based on the selected date is implemented.

## Conclusion

In this exercise, you explored two examples of unit tests that can be used to check that your web app works correctly. In the event that the tests produce errors or unexpected outcomes, you'll have a better idea of where exactly you need to apply a fix.