Task 2

Design the Sextant Frontend

Tutorial: Intro to React

This tutorial doesn't assume any existing React knowledge.

Before We Start the Tutorial

We will build a small game during this tutorial. **You might be tempted to skip it because you're not building games — but give it a chance.** The techniques you'll learn in the tutorial are fundamental to building any React app, and mastering it will give you a deep understanding of React.

Tip

This tutorial is designed for people who prefer to **learn by doing**. If you prefer learning concepts from the ground up, check out our step-by-step guide. You might find this tutorial and the guide complementary to each other.

The tutorial is divided into several sections:

- Setup for the Tutorial will give you **a starting point** to follow the tutorial.
- Overview will teach you **the fundamentals** of React: components, props, and state.
- Completing the Game will teach you **the most common techniques** in React development.
- Adding Time Travel will give you **a deeper insight** into the unique strengths of React.

You don't have to complete all of the sections at once to get the value out of this tutorial. Try to get as far as you can — even if it's one or two sections.

What Are We Building?

In this tutorial, we'll show how to build an interactive tic-tac-toe game with React.

You can see what we'll be building here: **Final Result**. If the code doesn't make sense to you, or if you are unfamiliar with the code's syntax, don't worry! The goal of this tutorial is to help you understand React and its syntax.

We recommend that you check out the tic-tac-toe game before continuing with the tutorial. One of the features that you'll notice is that there is a numbered list

to the right of the game's board. This list gives you a history of all of the moves that have occurred in the game, and it is updated as the game progresses.

You can close the tic-tac-toe game once you're familiar with it. We'll be starting from a simpler template in this tutorial. Our next step is to set you up so that you can start building the game.

Prerequisites

We'll assume that you have some familiarity with HTML and JavaScript, but you should be able to follow along even if you're coming from a different programming language. We'll also assume that you're familiar with programming concepts like functions, objects, arrays, and to a lesser extent, classes.
If you need to review JavaScript, we recommend reading this guide. Note that we're also using some features from ES6 — a recent version of JavaScript. In this tutorial, we're using arrow functions, classes, `let`, and `const` statements. You can use the Babel REPL to check what ES6 code compiles to.

Setup for the Tutorial

There are two ways to complete this tutorial: you can either write the code in your browser, or you can set up a local development environment on your computer.

Setup Option 1: Write Code in the Browser

This is the quickest way to get started!
First, open this **Starter Code** in a new tab. The new tab should display an empty tic-tac-toe game board and React code. We will be editing the React code in this tutorial.
You can now skip the second setup option, and go to the Overview section to get an overview of React.

Setup Option 2: Local Development Environment

This is completely optional and not required for this tutorial!

Optional: Instructions for following along locally using your preferred text editor

1.
2.
3.
4.
5.
6.

Help, I'm Stuck!

If you get stuck, check out the community support resources. In particular, Reactiflux Chat is a great way to get help quickly. If you don't receive an answer, or if you remain stuck, please file an issue, and we'll help you out.

Overview

Now that you're set up, let's get an overview of React!

What Is React?

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called "components".
React has a few different kinds of components, but we'll start with `React.Component` subclasses:

```
class ShoppingList extends React.Component {
  render() {
    return (
      <div className="shopping-list">
        <h1>Shopping List for {this.props.name}</h1>
        <ul>
          <li>Instagram</li>
          <li>WhatsApp</li>
          <li>Oculus</li>
        </ul>
      </div>
    );
  }
}

// Example usage: <ShoppingList name="Mark" />
```

We'll get to the funny XML-like tags soon. We use components to tell React what we want to see on the screen. When our data changes, React will efficiently update and re-render our components.
Here, ShoppingList is a **React component class**, or **React component type**. A component takes in parameters, called `props` (short for "properties"), and returns a hierarchy of views to display via the `render` method.