TASK

State Management and Component Lifecycle

Visit our website

Introduction

WELCOME TO THE STATE MANAGEMENT AND COMPONENT LIFECYCLE TASK!

Your React apps are becoming more useful with each task. You are now able to add attractive components to your UI. However, for your React apps to be truly responsive, your components need to be able to react to events. We will discuss how to do this in this task. We will also learn how to use React-Router to render and display different web pages.

REACT EVENTS

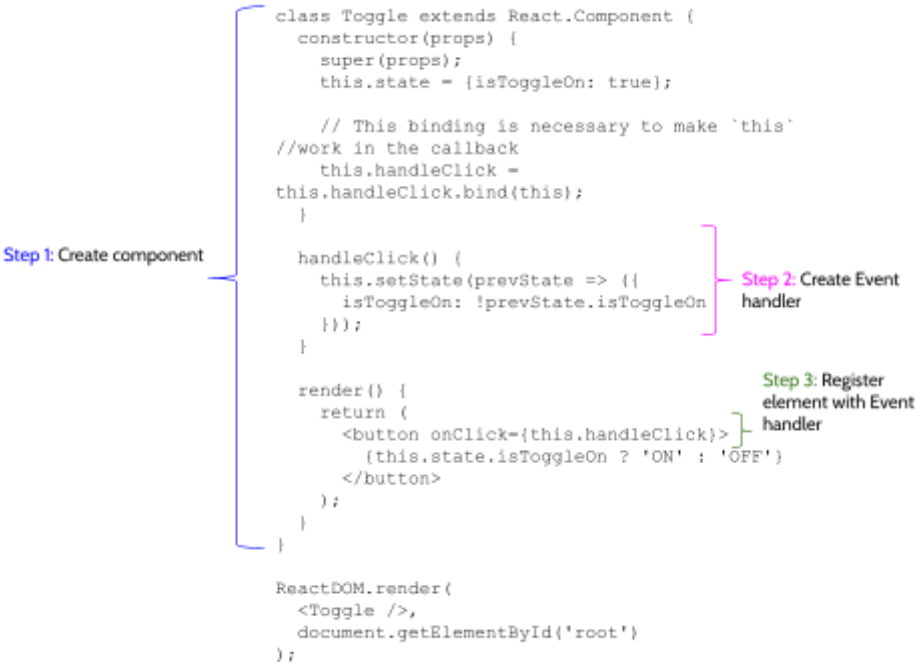
The concept of event-driven programming should not be new to you. We have already created several web applications that have responded to events. Unsurprisingly, React applications are event-driven as well. In this section, you will learn how to handle events with React.

React supports a host of events. You can find a list of these events **here**.

To create a component that responds to a specific event you must:

1. Create a class component (as shown in the previous task).
2. Create an event handler that responds to the event.
3. Register the React component with the event handler.

Consider the following **example** which illustrates how this is done:



Here are some important points to note regarding handling events with React:

- Use camelCase notation to name React events.
- As shown in the example above, with JSX you must pass a function (in the curly braces `{}`) as an event handler. Do not actually call the function here — notice the lack of parentheses after the function name.
- Consider the code below "`//work in the callback`" in the above example. Notice how one has to bind `this.handleClick` and pass it to `onClick` (see Step 3) so that this is properly defined when the function is called. See **here** for an alternative way of implementing this functionality.

REACT-ROUTER

Often, you are going to need to dynamically display different components based on a specific URL. To do this, we use React-router which is a library of code that matches a URL with components. There are two subsets of React-router, react-router-dom (used for web apps) and react-router-native (used for mobile apps).

To use react-router, do the following:

Step 1: install react-router-dom by typing
"`npm install --save react-router-dom@latest`" in the command line interface.

Step 2: Open src/index.js and import { BrowserRouter } from 'react-router-dom';

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { BrowserRouter } from "react-router-dom";

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
reportWebVitals();
```

The BrowserRouter object is what actually changes the components based on the URL. It is the brains behind what appears. BrowserRouter can have at most only one child.

Step 3: Open src/App.js and replace the default markup with some routes as soon below

```
import { Routes, Route, Link  } from "react-router-dom";
function App() {
  return (
    <div className="App">
      <Header />
      <Routes>
        <Route exact path="/" element={<Landing />} />
        <Route path="/cart" element={<ShoppingCart />} />
      </Routes>
    </div>
  );
}
// ADD OTHER COMPONENTS HERE
export default App;
```

Now, still in src/App.js, create your route components

```
// App.js
function Landing() {
  return (
    <div>
      <main>
        <h2>Welcome to the homepage!</h2>
      </main>
      <nav>
        <Link to="/cart">Cart</Link>
      </nav>
    </div>
  );
}

function ShoppingCart() {
  return (
    <div>
      <main>
        <h2>Shopping Cart</h2>
      </main>
      <nav>
```

```
        <Link to="/">Home</Link>
      </nav>
    </div>
  );
}

function Header() {
  return (
    <h1>
     <u>This is the Header</u>
    </h1>
  );
}
```

Step 4: Create rules using Route. The rules describe which components will be visible based on routes/URLs. For example, in the code above:

- `<Route path="/cart" element={<ShoppingCart />}/>` would display the ShoppingCart component for all URLs that contain the string "/Cart".
- `<Route exact path="/" element={<Landing />}/>` would display the Landing component for the root URL, e.g. **http://hyperiondev.com/**
    - The code `exact` specifies that the URL must be exactly the one specified, not just **contain** that value.
    - If `exact` were to be removed from the code in the example above, the URL "cart" would also display the Landing component because that URL **also contains the string "/"**.
    - The `Link` component provides declarative, accessible navigation around your application. E.g `<Link to="/">Home</Link>` will take use to the Landing component when Home is clicked on.

Note: Components that you want to display on all pages regardless of the URL can be added without a Route tag. For example, the Header component will be displayed on all pages, regardless of the URL.

For more information about using React Router, see **here** and watch **these videos**.

Compulsory Task 1

Follow these steps:

- Create a navigation menu component for the website for the fictitious company you have been working on in the last tasks. This component should include menu items for the home page, user profile page, shopping page, and legal page.

- Modify the **App.js** file you created in your **previous task** to do the following:
    - Display the header component on every page.
    - Display the menu component on every page.
    - Create a "Logout" button on the user profile page which displays "User has logged out" in an alert when the button is clicked.
    - Only display the landing component on the home page (i.e. root URL - "/")
    - Display at least 3 product components when the "shop" menu item is selected.

Compulsory Task 2

Create an interest calculator for the shopping website above. Add it as its own component and page with its own menu item and URL path.

The page should ask the user for their shopping total, and the number of months over which they'd like to pay back said amount. Use a non-compounding interest rate of 20% to calculate their monthly payments as well as the total they will have paid over all the months.

Remember to submit everything **except** the **node_modules** directory.

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think the content of this task, or this course as a whole, can be improved, or think we've done a good job?

Click here to share your thoughts anonymously.

REFERENCE

React.js. (2022). Getting Started – React. Retrieved 13 May 2022, from **https://reactjs.org/docs/getting-started.html**

Tutorial v6.8.0. (n.d.). React Router. **https://reactrouter.com/en/6.8.0/start/tutorial**