Hyperiondev

TASK

React III: Events

Visit our website

Introduction

WELCOME TO THE REACT EVENTS 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 display certain components based on the URL specified by a user.

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to login to **www.hyperiondev.com/portal** to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

REACT EVENTS

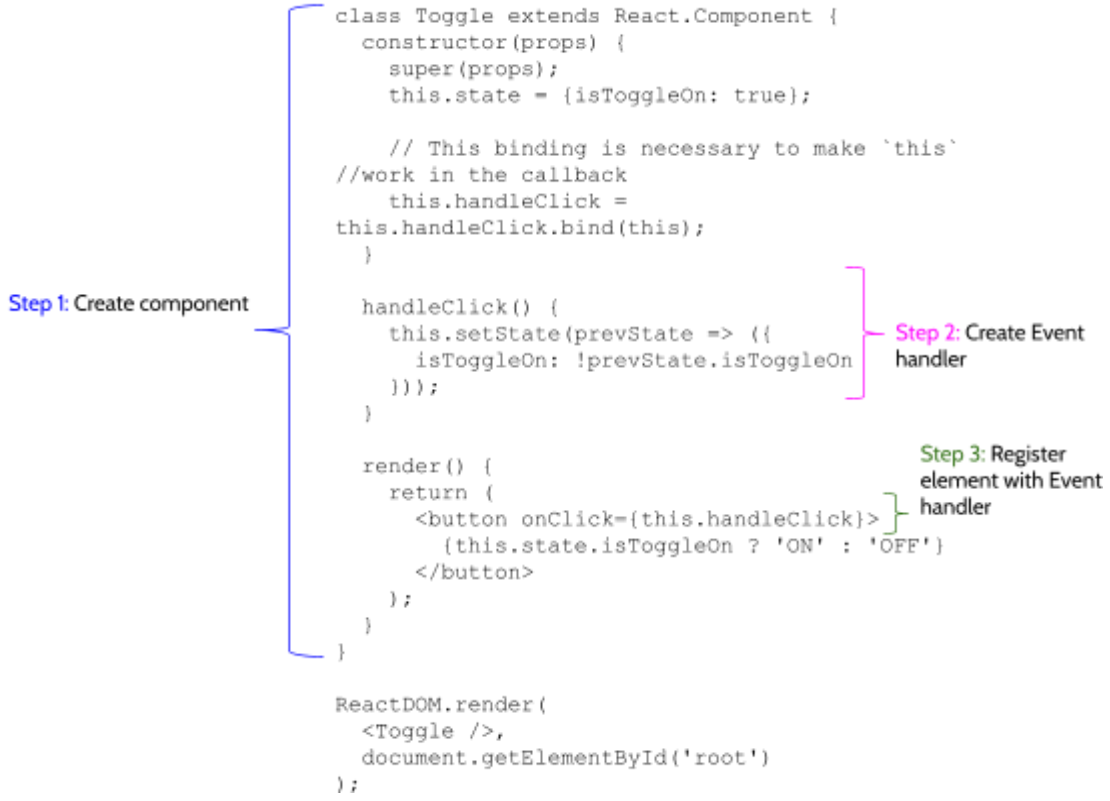
The concept of event-driven programming should not be new to you. We have already created several web applications that have responded to events. Unsurprisingly, React applications are event-driven as well. In this section, you will learn how to handle events with React.

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 the component (as shown in a 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:




```
class Toggle extends React.Component {
  constructor(props) {
    super(props);
    this.state = {isToggleOn: true};

    // This binding is necessary to make `this`
//work in the callback
    this.handleClick =
this.handleClick.bind(this);
  }

  handleClick() {
    this.setState(prevState => ({
      isToggleOn: !prevState.isToggleOn
    }));
  }

  render() {
    return (
      <button onClick={this.handleClick}>
        {this.state.isToggleOn ? 'ON' : 'OFF'}
      </button>
    );
  }
}

ReactDOM.render(
  <Toggle />,
  document.getElementById('root')
);
```

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 (use 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`" in the command line interface.

For all the steps that follow, refer to the code example below:

```
import React from 'react';
import { BrowserRouter, Route } from 'react-router-dom';

const Header = () => <h2>Header</h2>;
const Landing = () => <h2>Landing</h2>;
const ShoppingCart = () => <h2>Shopping Cart</h2>;

const App = () => {
   return (
       <div>
           <BrowserRouter>
               <div>
                   <Header />
                   <Route exact={true} path="/" component={Landing} />
                   <Route path="/Cart" component={ShoppingCart} />
               </div>
           </BrowserRouter>
       </div>
   );
};

export default App;
```

Step 2: `import {BrowserRouter, Route} from 'react-router-dom';`

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.

The Route object is used to set up the rules.

Step 3: In the function that defines the App component, wrap all the other components in the BrowserRouter tag.

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" component={ShoppingCart}/>` would display the ShoppingCart component for all URLs that contain the string "/Cart".
- `<Route exact={true} path="/" component={Landing}/>` would display the Landing component for the root URL, e.g. **http://hyperiondev.com/**
    - The code `exact={true}` specifies that the URL must be exactly the one specified, not just **contain** that value.
    - If `exact={true}` 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 "/"**.

Note: Components that you want to display on all pages regardless of the URL can be added without a Route tag. For example, see line 14 in the code above. The Header component will be displayed on all pages, regardless of the URL.

For more information about using React Router, see **here** and watch **this video**.

SPOT CHECK 1
Let's see what you can remember from this section.

1. What are the steps to create a component that responds to a specific event?
2. What are the steps to use react-router?

Instructions

- The React related tasks would see you creating apps that need some modules to run. These modules are located in a folder called 'node_modules', which is created when you run the following command from your command line or terminal: 'npx create-react-app my-app-name' or similar. Please note that this folder typically contains hundreds of files which, if you're working directly from Dropbox, has the potential to ***slow down Dropbox sync and possibly your computer***. As a result, please follow this process when creating/running such apps:
    - Create the app *on your local machine* (outside of Dropbox) by following the instruction in the compulsory task.
    - When you're ready to have your mentor review the app, please *delete the node_modules folder*.
    - Compress the folder and upload it to Dropbox.

  Your mentor will, in turn, decompress the folder, install the necessary modules and run the app from their local machine.

Compulsory Task 1

Follow these steps:

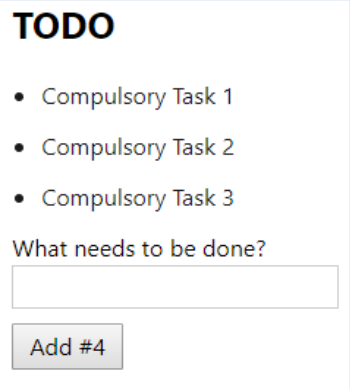
- Create a menu component for the website for the fictitious company you have been working on in the last task. This component should include a menu item that allows a user to view and change their profile. The component should also include a menu item that allows the user to "shop" and a menu item to return to the homepage.

- 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. The menu component should only display relevant items. For example, if the user is on the "shop" page, the "shop" menu item should no longer be displayed.
    - 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

Follow these steps:

- Create a simple to-do app using React. A user should be able to enter a number of items that get displayed as a to-do list in your app. The user should also be able to delete a specific item from the list.



Completed the task(s)?

Ask your mentor to review your work!

Review work

Rate us
Share your thoughts

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

Think that 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. (2020). Getting Started – React. Retrieved 6 August 2020, from https://reactjs.org/docs/getting-started.html

SPOT CHECK 1 ANSWERS

1.
    a. Create the component (as shown in a previous task).
    b. Create an event handler that responds to the event.
    c. Register the React component with the event handler.
2.
    a. Step 1: install react-router-dom by typing "`npm install --save react-router-dom`" in the command line interface.
    b. Step 2: `import {BrowserRouter, Route} from 'react-router-dom';`
    c. Step 3: In the function that defines the App component, wrap all the other components in the BrowserRouter tag.
    d. Step 4: Create rules using Route. The rules describe which components will be visible based on routes/URLs.