Active page

Last updated December 2023

Another everyday use case when an app contains a route is highlighting the currently active page in the menu.
Say you have a navigation menu that has two links:

1. **Home** that takes the user to /
2. **About** that takes the user to /about

Then it is possible to automatically highlight the **About** link with React Router when the user is on the /about route. Similarly, you can highlight the **Home** when the user is on the / route. All of that without having to write conditionals and complicated logic.
Here's how:

```
import {NavLink} from "react-router-dom";

function getClassName({isActive}) {
    if (isActive) {
        return "active"; // CSS class
    }
}

function App() {
    return <ul>
        <li>
            <NavLink to="/" className=
{getClassName}>Home</NavLink>
        </li>
        <li>
            <NavLink to="/about" className=
{getClassName}>About</NavLink>
        </li>
    </ul>
}
```

active page's menu link highlighted in bold.

```
.active {
    font-weight: bold;
}
```

The `className` prop in the `NavLink` component can also accept a function (so, not just a string). This function is automatically called by React Router with an argument, which is an object containing various properties about the navigation state. One of these properties is `isActive`. We're immediately destructuring that in the function parameter, which is why we have `{isActive}` in the parameter.

Then, we check if `isActive` is truthy. In that case, we return the name of the CSS class, `active`.

The `isActive` returns a boolean when the current React Router path is the same as the `NavLink`'s `to` attribute.

React Router will now automatically add the class `active` to the `<NavLink />` corresponding to the currently active route.

Shorter version

If you use the ternary operator, there's a shorter version to write the above. If you've taken the **Learn JavaScript** course, you may have noticed that we completely skipped the ternary operator. That's because it often negatively impacts readability. However, the following use case is considered acceptable.

The ternary operator allows you to replace an `if/else` statement using the `condition ? truthy expression : falsy expression`.

The following JavaScript:

```
function getClassName({isActive}) {
    if (isActive) {
        return "active";
```

```
    }
}
```

can be replaced with:

```
function getClassName({isActive}) {
    return isActive ? "active" : "";
}
```

You just have to make sure to put the `return` outside of the ternary. When `isActive` is truthy, the expression after ? will execute (`"active"`). Otherwise, the expression after the `:` will execute (`""`).

Knowing that, we can completely remove the separate function definition `getClassName` and write the `isActive ? "active" : ""` inside an arrow function `({isActive}) => isActive ? "active" : ""`.

If you find this too much of a jump, feel free to continue defining a separate function.

The final result looks like the below:

```
import {NavLink} from "react-router-dom";

function App() {
    return <ul>
        <li>
            <NavLink to="/" className={({isActive}) => isActive ?
"active" : ""}>Home</NavLink>
        </li>
        <li>
            <NavLink to="/about" className={({isActive}) =>
isActive ? "active" : ""}>About</NavLink>
        </li>
    </ul>
}
```

You may have noticed that we're using `NavLink` rather than `Link`.

This is **very** important as the `<Link />` component does **not** accept a function definition for the `className` prop.

A common mistake is to use the `ClassName` with a function definition on the `<Link />` component, but that won't work!

Recap

- The `className` prop, when used on a `NavLink`, accepts a function definition that lets you set the class based on `isActive` that is true when the current route matches the NavLink's `to` prop.

Was this helpful?