How Does Routing Work in React?

freecodecamp.org/learn/front-end-development-libraries-v9/lecture-routing-react-frameworks-and-dependency-management-tools/how-does-routing-work-in-react

In earlier lessons, you learned that React is a single page application. Single page applications are applications that contain one HTML file and use JavaScript to dynamically update any content on the page.

So what happens when you need to add multiple "pages" to your React application? How would you go about navigating to those different views?

Well, that is where React Router comes in.

React Router is a third party library that allows you to add routing to your React applications. To begin, you will need to install React Router in an existing React project like this:

```
npm i react-router
```

If you check the `package.json` file, you will see that `react-router` was added to the list of dependencies:

```
"dependencies": {
  "react": "^18.3.1",
  "react-dom": "^18.3.1",
  "react-router": "^7.2.0"
}
```

Then inside of your `main.jsx` or `index.jsx` file, you will need to import `BrowserRouter` and render `BrowserRouter` around your `App` component:

```
import { StrictMode } from "react";
import { createRoot } from "react-dom/client";
import { BrowserRouter } from "react-router";
import App from "./App.jsx";

import "./index.css";

createRoot(document.getElementById("root")).render(
  <StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </StrictMode>
);
```

To enable routes in your application, you will need to update your `import` statement to include the `Routes` and `Route` components like this:

```
import { BrowserRouter, Routes, Route } from "react-router";
```

Then inside of the `BrowserRouter`, add the `Routes` and `Route` components:

```
createRoot(document.getElementById("root")).render(
  <StrictMode>
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<App />} />
      </Routes>
    </BrowserRouter>
  </StrictMode>
);
```

The `path` and `element` are used to couple the URL and UI components together. In this case, we are setting up a route for the homepage that points to the `App` component.

It is common in larger applications to have multiple views and routes setup like this:

```
<Routes>
  <Route index element={<Home />} />
  <Route path="about" element={<About />} />

  <Route path="products">
    <Route index element={<ProductsHome />} />
    <Route path=":category" element={<Category />} />
    <Route path=":category/:productId" element={<ProductDetail />} />
    <Route path="trending" element={<Trending />} />
  </Route>
</Routes>
```

The `index` prop in these examples is meant to represent the default route for a given path segment. So the `Home` component will be shown at the root path (`/`) while the `ProductsHome` component will be shown at the `products` path.

You may have also noticed that we are nesting a few routes inside another route like this:

```
<Route path="products">
  <Route path="trending" element={<Trending />} />
</Route>
```

This means that the path of the child route will be appended to the parent route's path. So in this example, the `path` for the trending products will be `products/trending`.

If the `path` begins with a colon (`:`) then that represents a dynamic segment in the route:

```
<Route path=":category" element={<Category />} />
```

In this example we have a dynamic segment called `category`. When a user navigates to a URL like `products/brass-instruments`, then the view will change to the `Category` component and you can dynamically fetch the appropriate data based on the segment.

You can access the value of the dynamic segment by using the `useParams` hook inside the child component like this:

```
import { useParams } from "react-router";

export default function Category() {
  let params = useParams();
   {/* Accessing the category param: params.category */}
   {/* rest of code goes here */}
}
```

Dynamic routes are helpful because they allow you to create flexible and reusable components that can render different content based on the parameters in the URL. Instead of defining a fixed list of paths for every possible route, you can use dynamic segments to render various content based on what the user has requested.

Questions

What is React Router used for?

It is used to remove dynamic routes from your React applications.

It is used to setup static routes only for your React applications.

It is used to remove routes from your React applications.

It is used to add routing to your React applications.

Which hook is used to retrieve a dynamic parameter for a URL?

`usePar`

`useParams`

`useParameters`

`usePara`

Which of the following is the correct way to represent a dynamic segment in a path?

```
<Route path=":category" element={<Category
/>} />
```

```
<Route path="/category" element={<Category
/>} />
```

```
<Route path="\category" element={<Category
/>} />
```

```
<Route path="?category" element={<Category
/>} />
```