React Router Setup + Usage | Complete React Course in Hindi #16

In the last video, we have understood how to set the title and favicon in the React application. In this video, we are going to set up the React Router for our TextUtils application.

Why use React Router?

Till now, our application displays only one page in the home section. So we need a way to introduce multiple different pages or routes in our React application. The way we do so is by using React Router in our React application.

Install React Router package

Let’s see how to set up react-router in our code:

Procedure: The first thing we have to do is install the react-router package because it is not a part of the core react library. To do this, we need npm and a new terminal. So in the terminal, write the following code.

```
npm install react-router-dom
```

Once you have installed it, just go to the package.json file, and you will see the react-router-dom package right there.

Setting Up routing for our application

So now we have installed that package, how are we actually going to use and set up routing for our application.

Procedure: The first thing you have to do is go to the root component, that is, your app.js file, and import a few things from the react-router package. So let's do that first. We are going to import the Browser Router as Router, Switch, Route, and Link from *react-router-dom*. To import the following using the below command:

```
import {BrowserRouter as Router, Switch, Route, Link} from "react-router-dom"
```

1. **Using Router:** Now, we want to surround our entire application with the router component and what that means is we can use the router in the entire application. As a result, all components that are nested inside this app component(app.js) get access to the router. Hence, to surround your app with the router component, use:

```
<router></router>
```

2. **Using Switch:** The next step is to decide where we want our page content to go when we go to different pages. Well, I want to go inside the div, having className= 'container my-3', that is to 'about' and 'Textform' components. For this purpose, we are going to use the switch component (<switch></switch>). The switch component makes sure that only one route shows at any one time. All of our routes go inside this switch component.

3. **Using Route**: Alright, we need to set up our individual routes for the About and Home page. So, we will create a route for each page, for which we will be using the route component(<route></route>). At this moment, we have two pages of our application, and hence we are going to place two routes inside this switch component. But you can add more pages later on.

4. **Add property to Route**: Now, we are going to add the path property to the route. The path is basically the route, so for the home page, it would just be forward slash [‘/’].

For Home Page:

```
<Route path=“/”>
    <Textform showAlert={showAlert} heading="Enter the text to analyze below" mode={mo
<Route>
```

We have nested our component inside the route that we want to show when a user visits the forward-slash route. We would link to render our text form component whenever someone visits the forward-slash route.

For About Page:
Similarly, for the about page, it would be Forward slash about (/about).
Something like this:

```
<Route path= "/about">
    <About />
<Route>
```

Here, We have nested our about component to the /about the route.

Result: Alright, So now our route has been set up. If in your Navbar.js you have used the correct address of your Home and About page in your anchor tag, then your desired page will be rendered on clicking the button.
Problem: But wait a minute; in that case, your application will be sending the fresh request to the server each time, which we don't want to happen as it will reload the page each time, just as a normal application.

5. **Using 'Link to' tag:** To overcome this issue, instead of an Anchor tag, we will be using a special type of tag known as link tag. Let's go to the navbar and try to use a link tag instead of the anchor tag. The first thing we are going to do is import the link tag in Navbar.js as:

```
import { link } from 'react-router-dom';
```

Now we can replace the anchor tag with link to tag as follows:

```
<a href= "/">Home</a>
<a href= "/about">About</a>
```

Change the code to:

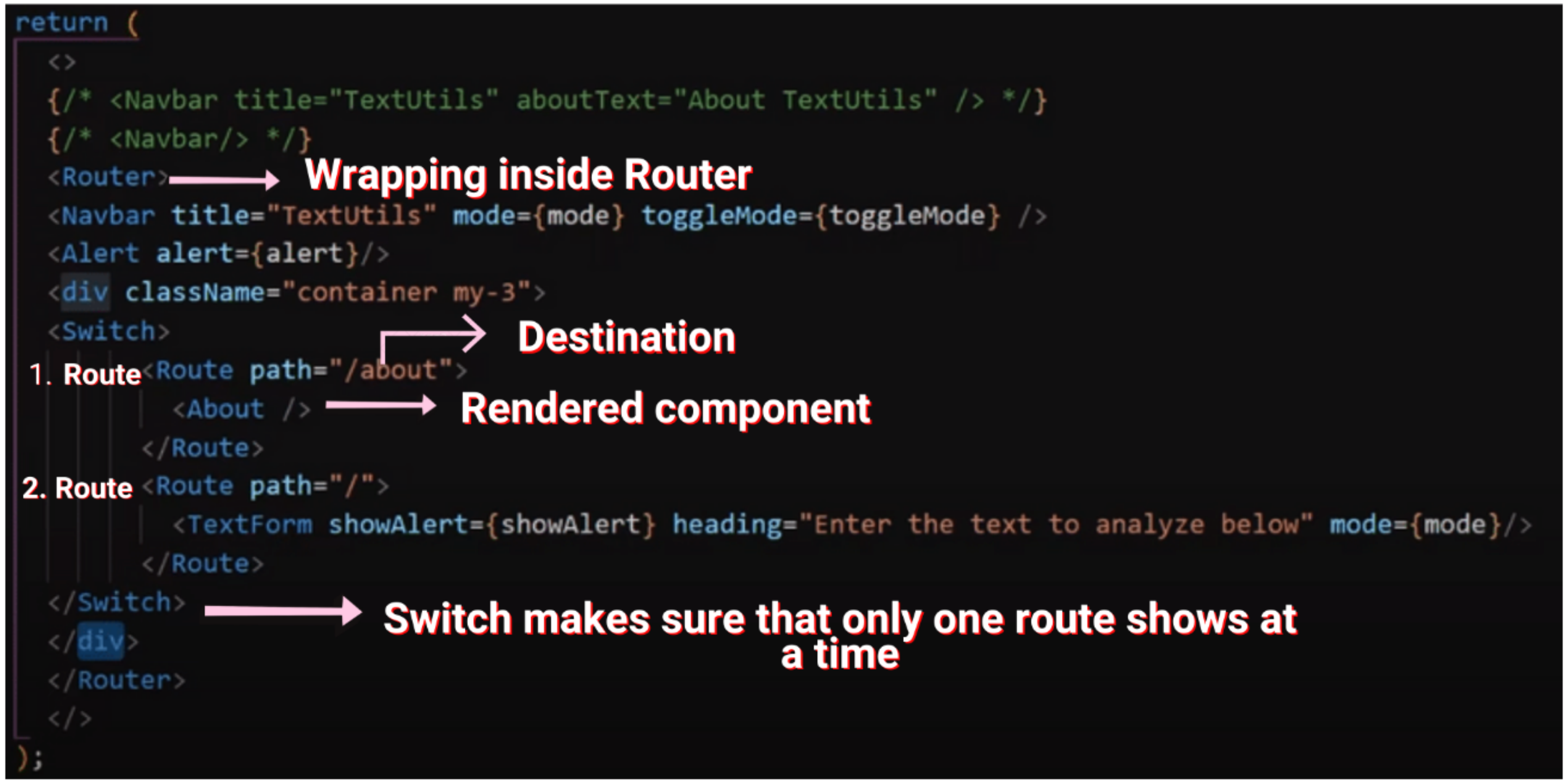
```
<Link to= "/">Home</Link>
<Link to= "/about">About</Link>
```

Result: You can now navigate between your About and Home page very quickly

Point to be Noted: The Navbar component of our application will always show up because it's not inside the switch statement. So it will be there for every single route.

In a Nutshell:

Here is the short summary of this tutorial:



```
return (
    <>
    {/* <Navbar title="TextUtils" aboutText="About TextUtils" /> */}
    {/* <Navbar/> */}
    <Router>
    <Navbar title="TextUtils" mode={mode} toggleMode={toggleMode} />
    <Alert alert={alert}/>
    <div className="container my-3">
    <Switch>
          <Route path="/about">
            <About />
          </Route>
          <Route path="/">
            <TextForm showAlert={showAlert} heading="Enter the text to analyze below" mode={mode}/>
          </Route>
    </Switch>
    </div>
    </Router>
    </>
);
```

Figure 1.1: A short summary

Remember you must use an ‘exact’ parameter with the Route component as it disables the partial matching of the route and makes sure that it only returns the route if the path is exact.