## **ROUTER**

### **Introduction**

In this lecture we are going to learn a little bit about routing in React. Up until now we have only written one-page applications. Once your application has multiple pages, you need to set up a reliable routing system. This is to handle the component or page that should be rendered when navigating to a certain route. For this we are going to use a package called react-router-dom.

Let’s go through it step by step. First of all, create a new project using create-react-app and get rid of the boilerplate code as you already learned in previous lessons. (Just leave the index.js and the App.js files)

Once you’ve done that, let’s create a new file called Profile.js inside the src directory, and add a basic functional component to it, which just contains an h1 element.

import React from "react"; const Profile = () => { return ( <div> <h1>Hello from Profile</h1> </div> ); }; export default Profile;

And also make sure your App.js file looks like this:

import React from "react"; const App = () => { return ( <div> <h1>Hello from App</h1> </div> ); }; export default App;

Once you have this, install the package. Open a terminal and run npm i react-router-dom. This is going to install all the dependencies for us. While this is happening, we can create a new file called Routes.js, which is going to be the file that handles all of our routes.

Once the package is finished installing, you can add this code to your Routes.js file:

import React from "react"; import { BrowserRouter, Switch, Route } from "react-router-dom"; import App from "./App"; import Profile from "./Profile"; const Routes = () => { return ( <BrowserRouter> <Switch> <Route path="/" component={App} /> <Route path="/profile" component={Profile} /> </Switch> </BrowserRouter> ); }; export default Routes;

So what is happening here? First we are importing React, our two components (Profile and App), and a few things from the package we just installed:

1. Route: Those are our routes with a path, which equals the url path, and a component that should be rendered when we navigate to this url.
2. BrowserRouter: Is a router, which uses the history API (pushState, replaceState and the popstate event) to keep your UI in sync with the URL. For completion I have to mention that there are other options than BrowserRouter, but for your current projects you can assume that BrowserRouter is at the root of all your projects.
3. Switch: Renders the first child Route that matches the location. In other words, the Switch component is going to look through all your Routes and checks their path. The first Route, whose path matches the url will be rendered; all others will be ignored. Important to note is that in our above example, both routes; the homepage (“/”) as well as the profile route (“/profile”) contain, at first, a / in their path. This means that when you go to “/profile” you will see the App component render, as it is the first path that matches the url. Our profile route gets ignored because it is second place. Therefore, our Profile component never gets rendered.

Let’s check this behavior in the browser for better understanding. But before we do that we have to do one more thing: change our index.js file. This is because we don’t want our App.js file to be the first file to be called when our application runs. Instead, we want our Routes.js to be the first. Your index.js should look something like this:

import React from "react"; import ReactDOM from "react-dom"; import Routes from "./Routes"; ReactDOM.render( <React.StrictMode> <Routes /> </React.StrictMode>, document.getElementById("root") );

Once this is done, go ahead and run npm start and check out both routes: the home route “/” and the profile route “/profile”. In both cases, the “Hello from App” from our App component is displayed.

There are two solutions to this problem.

1. Order your Routes by specificity from top to bottom. Change the order of your two routes so that you have the route with the path equal to “/profile” on top of the homepage. If you check it out in the browser now, you should see the App component on the home route and the Profile component on the “/profile” route. The problem with this solution is that it might take a lot of work sorting them once your application has several routes.
2. Add the exact keyword to your routes, like so:

<Route exact path="/" component={App} /> <Route exact path="/profile" component={Profile} />

The exact keyword specifies that the route path has to match the URL path exactly, as opposed to finding the first character (or string of characters) in the routes list that matches the URL path.

You should now have enough basics to get started with React routing. There are a lot more features to react-router-dom which are extremely useful, but out of the scope of this lesson. If you are interested in learning some more, we recommend you look into the history or match object. Definitely go and check out more advanced concepts once you are familiar with the basics.

### **Assignment**

1. Go and add a few new routes to the application we created above; playing around with it is the best practice. Consider deleting it completely and rewriting it for practice.
2. Browse a little bit through the React Router [documentation](https://reacttraining.com/react-router/web/guides/quick-start). You don’t need to read through all of it, nor understand all of it. Just browse through the concepts we discussed here and re-read them. This is a great resource to refer back to.
3. Watch [this](https://www.youtube.com/watch?v=Law7wfdg_ls) video for reviewing. You should already be familiar with those concepts.

### **Additional Resources**

* [This video by The Net Ninja](https://www.youtube.com/watch?v=QUz3k2O3ZJU&ab_channel=TheNetNinja) is an excellent review of what we learned here.
* For some extra practice/review, check out [this article](https://css-tricks.com/learning-react-router/).