React. Lesson 5

Routing

Installation

yarn add react-router-dom

npm install react-router-dom

BrowserRouter addition

- Creating browserRouter
- Choosing starting element

src/main.jsx

```jsx
import * as React from "react";
import * as ReactDOM from "react-dom/client";
import {
  createBrowserRouter,
  RouterProvider,
} from "react-router-dom";
import "./index.css";

const router = createBrowserRouter([
  {
    path: "/",
    element: <div>Hello world!</div>,
  },
]);

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <RouterProvider router={router} />
  </React.StrictMode>
);
```

BrowserRouter addition

src/main.jsx

```
/* existing imports */
import Root from "./routes/root";

const router = createBrowserRouter([
  {
    path: "/",
    element: <Root />,
  },
]);

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <RouterProvider router={router} />
  </React.StrictMode>
);
```

Error handling

- Usage of useRouterError() hook to access error attributes

src/error-page.jsx

```jsx
import { useRouteError } from "react-router-dom";

export default function ErrorPage() {
  const error = useRouteError();
  console.error(error);

  return (
    <div id="error-page">
      <h1>Oops!</h1>
      <p>Sorry, an unexpected error has occurred.</p>
      <p>
        <i>{error.statusText || error.message}</i>
      </p>
    </div>
  );
}
```

Error handling

- Adding errorElement element to router

src/main.jsx

```jsx
/* previous imports */
import ErrorPage from "./error-page";

const router = createBrowserRouter([
  {
    path: "/",
    element: <Root />,
    errorElement: <ErrorPage />,
  },
]);

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <RouterProvider router={router} />
  </React.StrictMode>
);
```

Path with parameters

- :contactId works as a dynamic parameters to access in routes

src/main.jsx

```
/* existing imports */
import Contact from "./routes/contact";

const router = createBrowserRouter([
  {
    path: "/",
    element: <Root />,
    errorElement: <ErrorPage />,
  },
  {
    path: "contacts/:contactId",
    element: <Contact />,
  },
]);

/* existing code */
```

Child routes

- By assigning route as a child we can access it using parent URL + it’s own URL

src/main.jsx

```jsx
const router = createBrowserRouter([
  {
    path: "/",
    element: <Root />,
    errorElement: <ErrorPage />,
    children: [
      {
        path: "contacts/:contactId",
        element: <Contact />,
      },
    ],
  },
]);
```

Outlet

- Outlet lets us to render child route elements as a specific tag

src/routes/root.jsx

```jsx
import { Outlet } from "react-router-dom";

export default function Root() {
  return (
    <>
      {/* all the other elements */}
      <div id="detail">
        <Outlet />
      </div>
    </>
  );
}
```

Link

- Link component lets us to navigate to a specific path

src/routes/root.jsx

```jsx
import { Outlet, Link } from "react-router-dom";

export default function Root() {
  return (
    <>
      <div id="sidebar">
        {/* other elements */}

        <nav>
          <ul>
            <li>
              <Link to={`contacts/1`}>Your Name</Link>
            </li>
            <li>
              <Link to={`contacts/2`}>Your Friend</Link>
            </li>
          </ul>
        </nav>

        {/* other elements */}
      </div>
    </>
  );
}
```

Loading data in routes

- We need to create a loader function

src/routes/root.jsx

```jsx
import { Outlet, Link } from "react-router-dom";
import { getContacts } from "../contacts";

export async function loader() {
  const contacts = await getContacts();
  return { contacts };
}
```

Loading data in routes

- Then, we import loader and assign it to loader property of route

src/main.jsx

```jsx
/* other imports */
import Root, { loader as rootLoader } from "./routes/root";

const router = createBrowserRouter([
  {
    path: "/",
    element: <Root />,
    errorElement: <ErrorPage />,
    loader: rootLoader,
    children: [
      {
        path: "contacts/:contactId",
        element: <Contact />,
      },
    ],
  },
]);
```

Loading data in routes

- Finally, we can access data using useLoaderData() hook

src/routes/root.jsx

```jsx
import {
  Outlet,
  Link,
  useLoaderData,
} from "react-router-dom";
import { getContacts } from "../contacts";

/* other code */

export default function Root() {
  const { contacts } = useLoaderData();
```

Parameters

- Loader can accept an object with attribute params, which will contain all the named values we identify in route's path

src/routes/contact.jsx

```jsx
import { Form, useLoaderData } from "react-router-dom";
import { getContact } from "../contacts";

export async function loader({ params }) {
  const contact = await getContact(params.contactId);
  return { contact };
}

export default function Contact() {
  const { contact } = useLoaderData();
  // existing code
}
```

Parameters

- We can access parameters using useParams() hook

```
import * as React from 'react';
import { Routes, Route, useParams } from 'react-router-dom';

function ProfilePage() {
  // Get the userId param from the URL.
  let { userId } = useParams();
  // ...
}

function App() {
  return (
    <Routes>
      <Route path="users">
        <Route path=":userId" element={<ProfilePage />} />
        <Route path="me" element={...} />
      </Route>
    </Routes>
  );
}
```