Лекция 9
Создание и управление навигацией

1. Создание и управление навигацией

После настройки базовой маршрутизации, реализации отложенной загрузки и добавления резервных компонентов, следующим важным шагом является предоставление пользователям средств для навигации по страницам нашего приложения.

React Router предоставляет компоненты **Link** и **NavLink,** аналогичные HTML- тегу **<a>.** Эти компоненты позволяют обновлять URL-адрес в адресной строке браузера без перезагрузки всей страницы.

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

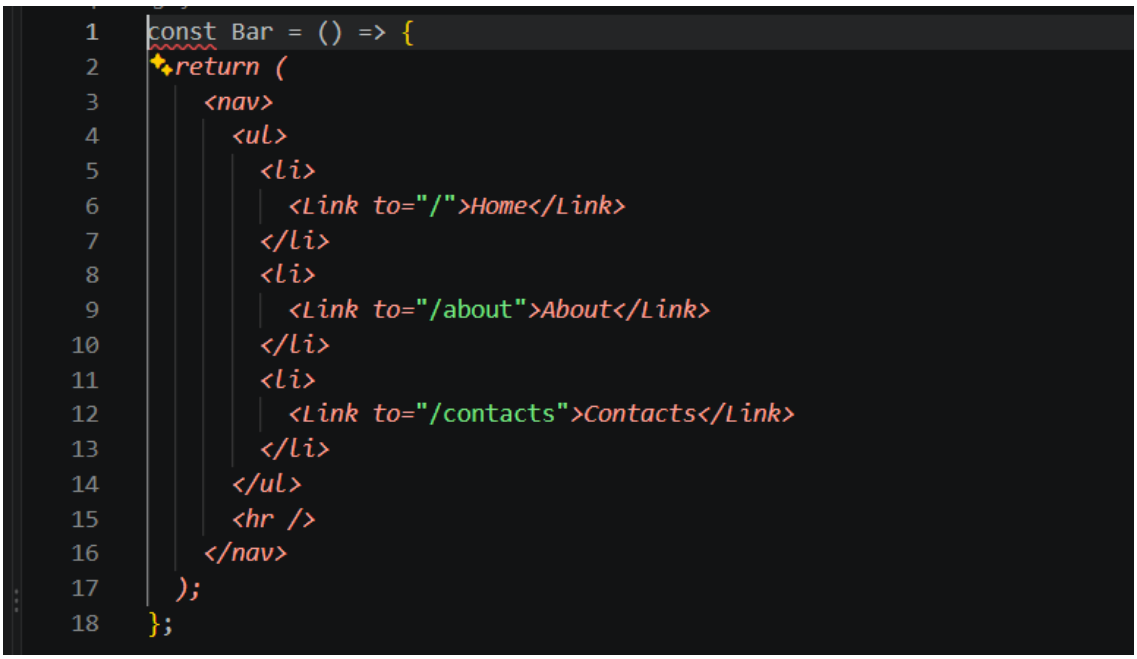
<Link to="path_value">Any text</Link>
```

– Этот **to** параметр указывает целевой маршрут или путь, на который должна вести ссылка. Замените **path_value** на путь, на который вы хотите перенаправить ссылку;

– **Any text** Это содержимое, которое будет отображаться в качестве ссылки. Вы можете заменить его желаемым текстом или содержимым JSX.

Пример. Давайте создадим отдельный компонент под названием Bar для управления навигацией всего приложения. Вот код:

```
const Bar = () => {
  return (
    <nav>
      <ul>
        <li>
          <Link to="/">Home</Link>
        </li>
        <li>
          <Link to="/about">About</Link>
        </li>
        <li>
          <Link to="/contacts">Contacts</Link>
        </li>
      </ul>
      <hr />
    </nav>
  );
};
```



```
1  const Bar = () => {
2    return (
3      <nav>
4        <ul>
5          <li>
6            <Link to="/">Home</Link>
7          </li>
8          <li>
9            <Link to="/about">About</Link>
10         </li>
11         <li>
12           <Link to="/contacts">Contacts</Link>
13         </li>
14       </ul>
15       <hr />
16     </nav>
17   );
18 };
```

Этот код определяет функциональный компонент с именем **Bar**, который представляет собой панель навигации. Он отображает **<nav>**элемент, содержащий неупорядоченный список <**ul**>с тремя элементами **<li>.** Каждый элемент списка включает **Link** компонент из **react-router-dom** библиотеки.

Строка 6: Первый **Link** компонент представляет собой ссылку на главную страницу;

Строка 9: Второй **Link** компонент представляет собой ссылку на страницу «О нас»;

Строка 12: Третий **Link** компонент представляет собой ссылку на страницу «Контакты».

Эти **Link** компоненты обеспечивают навигацию внутри приложения, обновляя URL-адрес в адресной строке браузера без перезагрузки страницы. Когда пользователь щелкает по ссылке, активируется соответствующий маршрут или путь, и отображается связанный с ним компонент.

2. Изучение полной логики React Router

Давайте внимательно изучим всё приложение, чтобы понять его функциональность. Предоставленная среда разработки Code Sandbox содержит полную логику React Router для этого приложения. Мы можем изучить и проанализировать её, чтобы понять, как используется React Router.

Нажмите «Открыть Sandbox», чтобы просмотреть код в **Code Sandbox**, при этом URL-адрес будет меняться по мере перехода по страницам.

index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App/App";
import { BrowserRouter } from "react-router-dom";

const root = ReactDOM.createRoot(document.getElementById("root"));
```

```
root.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```

App.jsx

```
import React, { Suspense, lazy } from "react";
import { Routes, Route } from "react-router-dom";
import Loader from "../Loader/Loader";
import Bar from "../Bar/Bar";
import Footer from "../Footer/Footer";
import "../index.css";

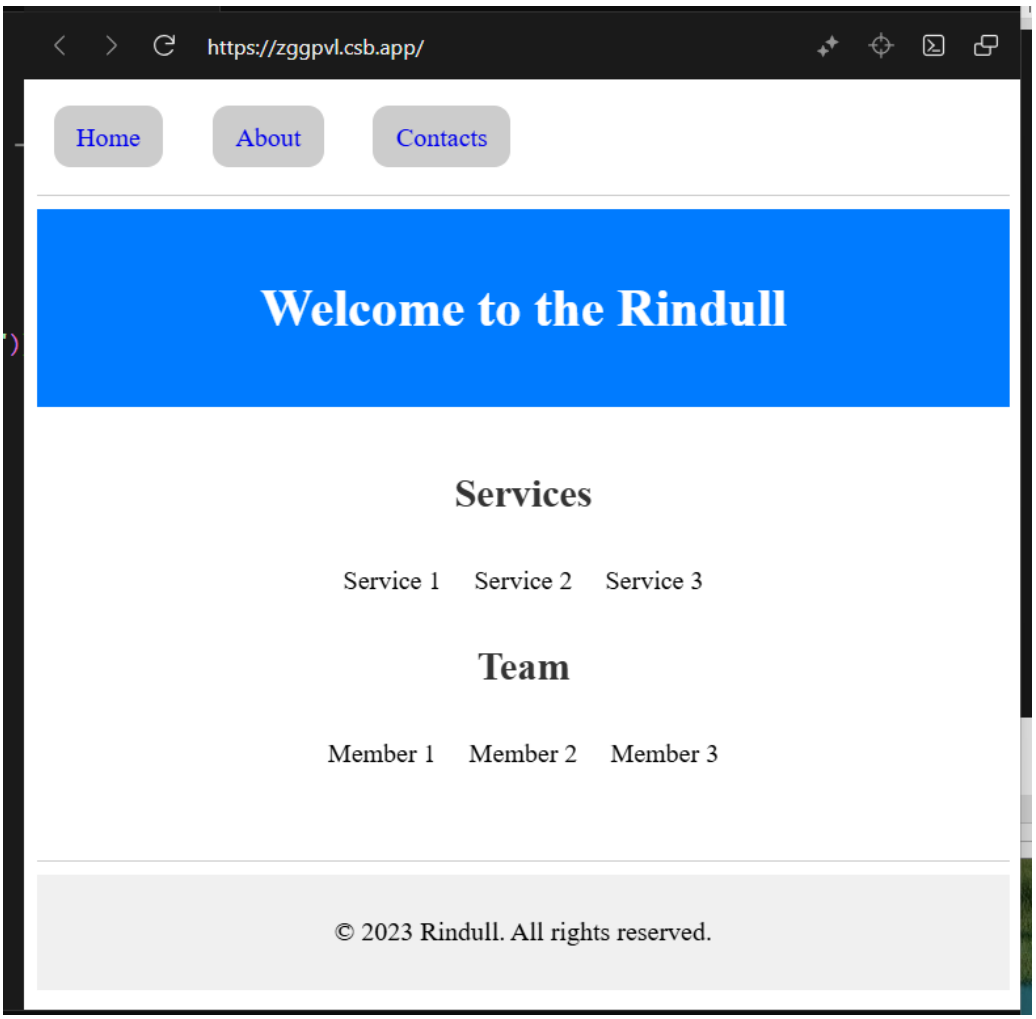
const HomePage = lazy(() => import("../HomePage/HomePage"));
const AboutPage = lazy(() => import("../AboutPage/AboutPage"));
const ContactsPage = lazy(() => import("../ContactsPage/ContactsPage"));

const App = () => {
  return (
    <>
      <Bar />
      <Suspense fallback={<Loader />}>
        <Routes>
          <Route path="/" element={<HomePage />} />
          <Route path="/about" element={<AboutPage />} />
          <Route path="/contacts" element={<ContactsPage />} />
        </Routes>
      </Suspense>
      <Footer />
    </>
  );
};

export default App;
```

Результат Code Sandbox, где URL меняется по мере навигации.

Ссылка: https://codesandbox.io/p/sandbox/bf065d18-9186-4184-983a-5938e44c9b27-zggpvl?file=%2Fsrc%2FApp%2FApp.jsx%3A7%2C1&from-embed



3. Краткий обзор основ React Router

Мы рассмотрели основные концепции React Router, важнейшей библиотеки для добавления функциональности маршрутизации в ваши React-приложения. Давайте подведем итоги рассмотренных ключевых моментов:

Установка. Первый шаг — установить библиотеку React Router в ваш проект. Это можно сделать, выполнив следующую команду в каталоге вашего проекта:

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

Импорт необходимых компонентов

Вам следует импортировать необходимые компоненты из **react-router-dom** пакета в файлы компонентов. К этим компонентам относятся **BrowserRouter, Route, Routes, Link, NavLink**. Эти компоненты являются строительными блоками для создания и управления маршрутами.

Настройка BrowserRouter

Для включения функции маршрутизации в вашем приложении необходимо обернуть корневой компонент в другой **BrowserRouter** компонент. Этот шаг инициализирует систему маршрутизации и позволяет осуществлять навигацию на основе URL-адреса.

Определение Routes маршрутов

Внутри **App** компонента (или соответствующего компонента) вы можете определить маршруты, используя компоненты **Route** и **Routes**. RouteКомпонент связывает определенный путь с компонентом, который будет отображаться, когда этот путь совпадает с текущим URL-адресом. Это определяет структуру и поведение навигации вашего приложения.

Ленивая загрузка с элементами Suspense

Для оптимизации производительности вашего приложения вы можете реализовать разделение кода и отложенную загрузку компонентов. Это достигается с помощью функции **lazy** из React и обертывания импортированного компонента в другой **Suspense** компонент. Отложенная загрузка гарантирует асинхронную загрузку компонентов по мере необходимости, что улучшает время первоначальной загрузки вашего приложения.

Создание ссылок

Вы можете создавать навигационные ссылки в своем приложении, используя компонент **Link** или **NavLink**. **Link** Компонент используется для перехода к определенному маршруту без полной перезагрузки страницы. Это обеспечивает более плавную и эффективную работу приложения для пользователей.