HyperionDev

Full-Stack Web Development

HyperionDev

Routing in React

Full Stack Web Development Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Full Stack Web Development Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: **www.hyperiondev.com/support**
- Report a **safeguarding** incident: **www.hyperiondev.com/safeguardreporting**
- We would love your **feedback** on lectures: **Feedback on Lectures**

Objectives

- Explain the purpose and benefits of routing in single-page applications.
- Implement routing in a React application using React Router.
- Navigate between different components using React Router.
- Create nested routes and use URL parameters to pass data.

Routing

Definition and Use Cases

- Routing can be termed as the **conditional rendering** of components based on the **URL** in the browser.
- Routing allows users to **navigate between different pages or views** within a web application.
- Routing with plain HTML/CSS used to be **file based**, the anchor (<a>) were used to create hyperlinks that link to different web pages which were the different (.html) files in your project.

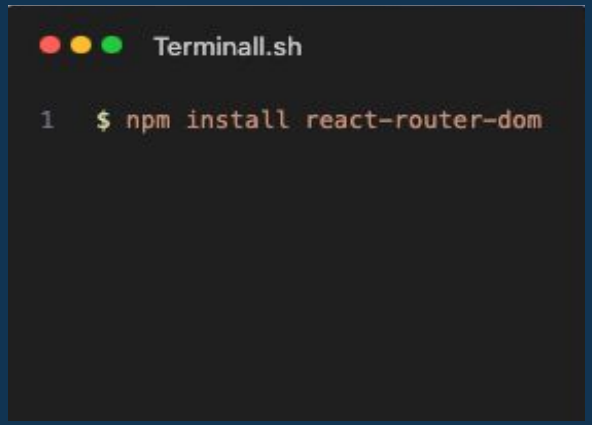
Routing in React

- In the context of React, **client side routing** is executed.
- This allows your app to **update the URL from a link click** without making another request for another document from the server, making your application **render immediately**.
- In simple terms, routing in React involves **dynamically updating the content** of the website without reloading the entire page.
- Routing in React is mostly implemented using **routing libraries** or frameworks. Two common libraries in use for a seamless routing experience are **React Router DOM** and **Reach Router**.

React Router DOM

Achieves client side routing in your React application by using its inbuilt routing APIs.

- To use React Router in your application, you need to install it first using npm or yarn



Configuration

- After installing React Router, you need to configure your app to use it. This will be done in the root of you Javascript file (index.js).

index.js

```
7   //other React imports
8   import { createBrowserRouter, RouterProvider } from 'react-router-dom';
9
10  const paths = createBrowserRouter([
11    {
12      path: '/',
13      element: <h1>Hello World</h1>
14    }
15  ])
16
17
18  const root = ReactDOM.createRoot(document.getElementById('root'));
19  root.render(
20    <React.StrictMode>
21      <RouterProvider router={paths} /> {/** replaced <App/> */}
22    </React.StrictMode>
23  );
```

React Router APIs

- From the configuration example shown, we made two important imports:
  1. **createBrowserRouter**: this configures Browser Router which enables client side Routing in our React application.
  - ➢ It is a function that takes in a list of available paths in our application, the paths will be defined by objects.
  - ➢ Currently, we've only created one path which is the home path using a '/' and it renders a <h1> text saying Hello World.

React Router APIs

2. **RouterProvider**: All path objects created by the createBrowserRouter API are passed to the provider component as a value of the router prop to render your app and enable routing.

- After this configuration, upon running your React server, you will have a text displaying Hello World on the home page.

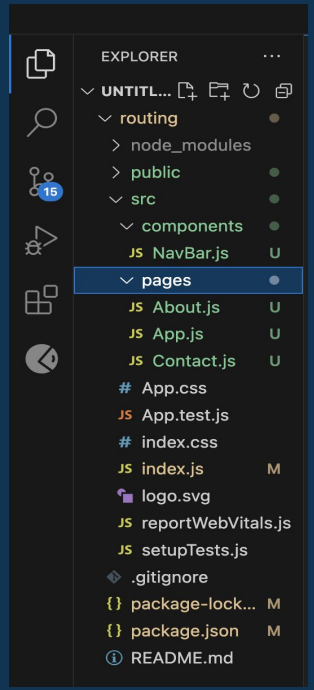
Multiple Pages

- Having multiple pages in our React app is one of the main achievements of routing.
- We do this by creating **other path objects** and **pointing** the path elements to their specific components.
- The **element property** of the path object will be replaced by a **React component from your project**.
- In this case, we have three components representing three pages and all are stored in a folder called pages for best practice purpose.

Multiple Pages

index.js

```
6   //other React imports
7   import App from './pages/App';
8   import About from './pages/About'
9   import Contact from './pages/Contact'
10  import { createBrowserRouter, RouterProvider } from 'react-router-dom';
11
12  const paths = createBrowserRouter([
13    {
14      path: '/',
15      element: <App/>
16    },
17    {
18      path: '/about',
19      element: <About/>
20    },
21    {
22      path: '/contact',
23      element: <Contact/>
24    }
25  ])
26
27
28  const root = ReactDOM.createRoot(document.getElementById('root'));
29  root.render(
30    <React.StrictMode>
31      <RouterProvider router={paths} />
32    </React.StrictMode>
33  );
```



Navigating through React Router pages

- For hyperlinks, we are used to utilizing the <a> tag in HTML. Using <a href=''> causes a page refresh which can lead to losing an application's state.
- To achieve complete client side routing with React Router, we use its <Link> element to navigate from page to page. Instead of the {href='/path'} attribute in <a> tags, the link element provides a { to='/path'} property to direct the link to the desired URL path.
- The <Link> element does not cause a page refresh hence the application's state cannot be lost.

Example

Note that the structure of the App component is also implemented on the About and Contact component

- The { Link } element is imported from ‘react-router-dom’
- You can also user { NavLink } to know whether a page is active or not.

NavBar.js

```
1   import { Link } from "react-router-dom"
2
3   const NavBar = () =>{
4       return (
5           <nav>
6               <Link to="/">Home</Link>
7               <Link to="/about">About</Link>
8               <Link to="/contact">Contact</Link>
9           </nav>
10      )
11  }
12
13  export default NavBar
```

App.js

```
1   import NavBar from "../components/NavBar"
2
3   function App () {
4       return (
5           <section>
6               <NavBar/>
7               <h1>Home</h1>
8           </section>
9       )
10  }
11
12  export default App
```

Dynamic Routing

- Dynamic routing is a way of rendering a new component by updating a particular segment in the URL called params.
- We achieve this by adding { :id } to the path, the colon section of the path will represent the dynamic segment. The suffix of the path will be replaced by respective path id or name.
- Note that you can name the id to anything as long as it rhymes with the intention. i.e { :itemId }, { :userId }

Example

index.js

```
6   //other React imports
7   import App from './pages/App';
8   import About from './pages/About'
9   import Contact from './pages/Contact'
10  import User from './pages/User';
11  import { createBrowserRouter, RouterProvider } from 'react-router-dom';
12
13
14  const paths = createBrowserRouter([
15    {
16      path: '/',
17      element: <App/>
18    },
19    {
20      path: '/about',
21      element: <About/>
22    },
23    {
24      path: '/contact',
25      element: <Contact/>
26    },
27    {
28      path: '/user/:userId', //dynamic path, has the /:userId suffix
29      element: <User/>
30    }
31  ])
32  //other configurations
```

NavBar.js

```
1   import { Link } from "react-router-dom"
2
3   const NavBar = () =>{
4       return (
5           <nav>
6               <Link to="/">Home</Link>
7               <Link to="/about">About</Link>
8               <Link to="/contact">Contact</Link>
9               <Link to="/user/1">User 1</Link>
10              <Link to="/user/2">User 2</Link>
11              <Link to="/user/3">User 3</Link>
12          </nav>
13      )
14  }
15
16  export default NavBar
```

Hyperiondev

Questions and Answers

HyperionDev

Thank You for attending!