Step8Up Academy

USING REACT ROUTER

FULL STACK SKILLS BOOTCAMP

MASTERING NAVIGATION IN REACT APPLICATIONS

- **Lesson Overview:**
- In this lesson, we will be introduced to:

1. What is React Router
2. Installation
3. Passing URL parameters
4. Using Query String paramaters

WHAT IS REACT ROUTER?

- **Why Use It?**

  Facilitates seamless navigation between views or components.

- Maintains application state across navigation.
- Enables deep linking and client-side routing.

WHAT IS REACT ROUTER?

- **Core Features:**

  Declarative routing.

- Nested and dynamic routes.
- Code-splitting for performance.

INSTALLATION

- Open your project directory in the terminal.
  Run the command:

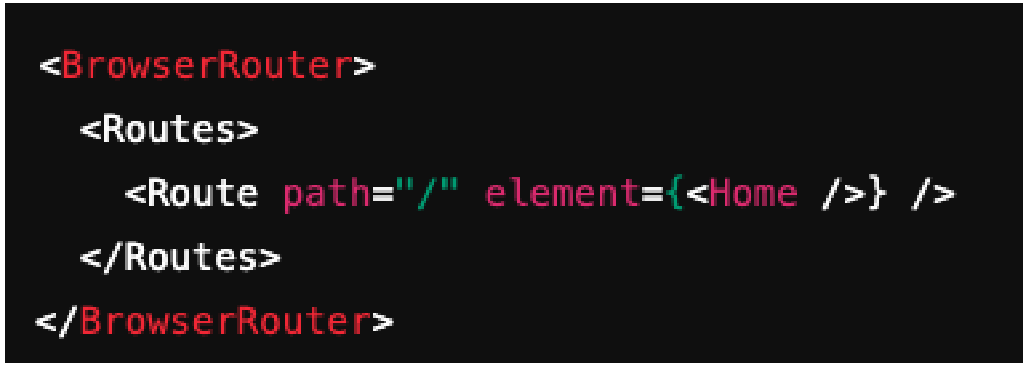
```
npm install react-router-dom
```

Import React Router components in your application

```
import { BrowserRouter, Route, Routes } from 'react-router-dom';
```

TYPES OF ROUTING

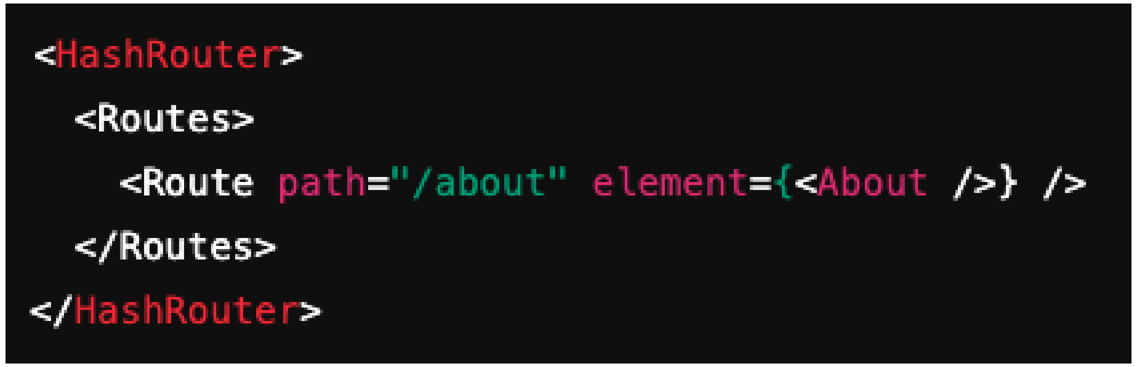
- Uses the HTML5 history API.



```
<BrowserRouter>
  <Routes>
    <Route path="/" element={<Home />} />
  </Routes>
</BrowserRouter>
```

TYPES OF ROUTING

- **HashRouter**
- Uses URL hash for routing.



```
<HashRouter>
  <Routes>
    <Route path="/about" element={<About />} />
  </Routes>
</HashRouter>
```

PASSING URL PARAMETERS

- Dynamic paths to pass parameters:

```
<Route path="/user/:id" element={<UserProfile />} />
```

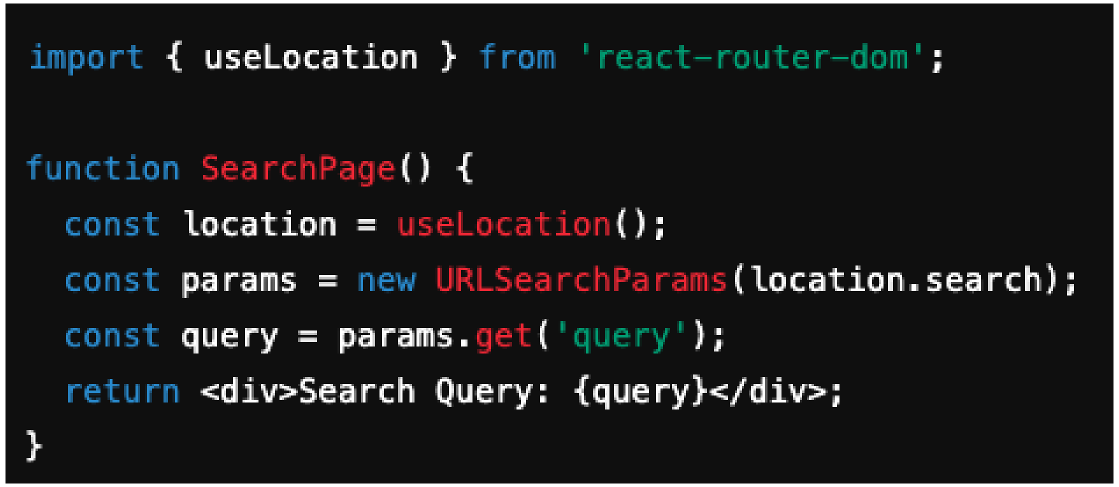
- Access parameters using useParams hook:

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

function UserProfile() {
  const { id } = useParams();
  return <div>User ID: {id}</div>;
}
```

USING QUERY STRING PARAMETERS

- Query strings allow passing optional parameters in the URL.
  Example:
  http://example.com/search?query=react
- Access query parameters using useLocation and URLSearchParams:



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

function SearchPage() {
  const location = useLocation();
  const params = new URLSearchParams(location.search);
  const query = params.get('query');
  return <div>Search Query: {query}</div>;
}
```

CONCLUSION

- **Key Takeaways:**
- React Router simplifies client-side routing.
- Supports dynamic routing with URL and query string parameters.
- Enhances user experience with declarative, efficient navigation.

CONCLUSION

- **Next Steps:**
  - Experiment with nested routes.
  - Explore route protection using useNavigate.
  - Combine with state management libraries like Redux.

Step8Up Academy

QUESTIONS?