Day 19 - NavLink, useNavigate & Programmatic Navigation

NavLink Component

What is NavLink?

`NavLink` is a special version of the `Link` component that knows whether it is "active" or not. This is useful for styling navigation menus.

Basic Usage

```jsx
import { NavLink } from 'react-router-dom';

<NavLink to="/home">Home</NavLink>
<NavLink to="/about">About</NavLink>
<NavLink to="/contact">Contact</NavLink>
```

Styling Active Links

Method 1: Using className

```jsx
<NavLink
  to="/home"
  className={({ isActive }) => isActive ? "active-link" : ""}
>
  Home
</NavLink>
```

Method 2: Using style

```jsx
<NavLink
  to="/about"
  style={({ isActive }) => ({
    color: isActive ? "red" : "blue",
    fontWeight: isActive ? "bold" : "normal"
  })}
>
  About
</NavLink>
```

Method 3: Using CSS (automatic .active class)

```jsx
// In your component
<NavLink to="/contact">Contact</NavLink>

// In your CSS
.active {
  color: red;
  font-weight: bold;
  text-decoration: underline;
}
```

NavLink Props

- `to` - The path to navigate to
- `className` - Function or string for CSS classes
- `style` - Function or object for inline styles
- `end` - Only considers link active when exact match
- `caseSensitive` - Makes matching case-sensitive

Example: Navigation Bar

```jsx
```

```jsx
import { NavLink } from 'react-router-dom';
import './navbar.css';

function Navbar() {
  return (
    <nav>
      <NavLink
        to="/"
        end
        className={({ isActive }) => isActive ? "nav-link active" : "nav-link"}
      >
        Home
      </NavLink>

      <NavLink
        to="/products"
        className={({ isActive }) => isActive ? "nav-link active" : "nav-link"}
      >
        Products
      </NavLink>

      <NavLink
        to="/about"
        className={({ isActive }) => isActive ? "nav-link active" : "nav-link"}
      >
        About
      </NavLink>
    </nav>
  );
}
```

---

useNavigate Hook

What is useNavigate?

`useNavigate` is a hook that returns a function to navigate programmatically (in JavaScript code, not just through clicks).

Basic Usage

```
jsx
```

```jsx
import { useNavigate } from 'react-router-dom';

function MyComponent() {
  const navigate = useNavigate();

  const handleClick = () => {
    navigate('/home');
  };

  return <button onClick={handleClick}>Go Home</button>;
}
```

Navigate with State

```jsx
const navigate = useNavigate();

navigate('/profile', {
  state: { userId: 123, from: 'dashboard' }
});

// Access in the target component
import { useLocation } from 'react-router-dom';

function Profile() {
  const location = useLocation();
  console.log(location.state.userId); // 123
}
```

Navigate Backwards/Forwards

```jsx
const navigate = useNavigate();

// Go back one page
navigate(-1);

// Go forward one page
navigate(1);

// Go back two pages
navigate(-2);
```

Replace Current Entry

```jsx
// Replace current history entry (user can't go back to this page)
navigate('/login', { replace: true });
```

Programmatic Navigation Use Cases

1. After Form Submission

```jsx
function LoginForm() {
  const navigate = useNavigate();

  const handleSubmit = async (e) => {
    e.preventDefault();

    const success = await loginUser(formData);

    if (success) {
      navigate('/dashboard');
    }
  };

  return <form onSubmit={handleSubmit}>...</form>;
}
```

2. Authentication Redirect

```jsx
```

```jsx
function ProtectedComponent() {
  const navigate = useNavigate();
  const isAuthenticated = checkAuth();

  useEffect(() => {
    if (!isAuthenticated) {
      navigate('/login', { replace: true });
    }
  }, [isAuthenticated, navigate]);

  return <div>Protected Content</div>;
}
```

3. Timed Redirect

jsx

```jsx
function SuccessPage() {
  const navigate = useNavigate();

  useEffect(() => {
    const timer = setTimeout(() => {
      navigate('/home');
    }, 3000);

    return () => clearTimeout(timer);
  }, [navigate]);

  return <div>Success! Redirecting in 3 seconds...</div>;
}
```

4. Conditional Navigation

jsx

```
function Checkout() {
  const navigate = useNavigate();

  const handleCheckout = () => {
    if (cartItems.length === 0) {
      navigate('/products');
    } else if (!isLoggedIn) {
      navigate('/login', { state: { from: '/checkout' } });
    } else {
      navigate('/payment');
    }
  };

  return <button onClick={handleCheckout}>Proceed</button>;
}
```

Differences: Link vs NavLink vs useNavigate

Feature	Link	NavLink	useNavigate
Purpose	Basic navigation	Navigation with active state	Programmatic navigation
Usage	Click to navigate	Click to navigate	Navigate in JS code
Active State	No	Yes	No
Use Case	Regular links	Navigation menus	After events/logic

Complete Example

```
jsx
```

```
import { BrowserRouter, Routes, Route, NavLink, useNavigate } from 'react-router-dom';
import './App.css';

function Navbar() {
  return (
    <nav className="navbar">
      <NavLink
        to="/"
        end
        className={({ isActive }) => isActive ? "active" : ""}
      >
        Home
      </NavLink>
      <NavLink
        to="/products"
        className={({ isActive }) => isActive ? "active" : ""}
      >
        Products
      </NavLink>
      <NavLink
        to="/about"
        className={({ isActive }) => isActive ? "active" : ""}
      >
        About
      </NavLink>
    </nav>
  );
}

function Home() {
  const navigate = useNavigate();

  return (
    <div>
      <h1>Home Page</h1>
      <button onClick={() => navigate('/products')}>
        View Products
      </button>
    </div>
  );
}

function Products() {
  const navigate = useNavigate();

  const handleProductClick = (id) => {
    navigate(`/products/${id}`, { state: { from: 'products-list' } });
  };

  return (
    <div>
      <h1>Products</h1>
      <button onClick={() => navigate(-1)}>Go Back</button>
    </div>
  );
}

function About() {
  return <h1>About Page</h1>;
}

function App() {
  return (
    <BrowserRouter>
      <Navbar />
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/products" element={<Products />} />
        <Route path="/about" element={<About />} />
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

```
css
```

```css
/* App.css */
.navbar {
  display: flex;
  gap: 20px;
  padding: 20px;
  background-color: #333;
}

.navbar a {
  color: white;
  text-decoration: none;
  padding: 10px 15px;
}

.navbar a.active {
  background-color: #007bff;
  border-radius: 5px;
  font-weight: bold;
}
```

Key Takeaways

1. **NavLink** is for navigation menus where you need to show which page is active
2. **useNavigate** is for navigating based on logic, events, or conditions
3. NavLink automatically adds an `active` class when the route matches
4. Use `navigate(-1)` to go back, like a browser back button
5. You can pass state with navigation for data sharing between routes
6. Use `replace: true` when you don't want the user to go back to that page

---

Practice Exercise

Create a simple app with:

1. A navbar using NavLink with active styling
2. A login form that navigates to dashboard on success
3. A dashboard with a logout button that goes back to login
4. A products page with a back button using `navigate(-1)`