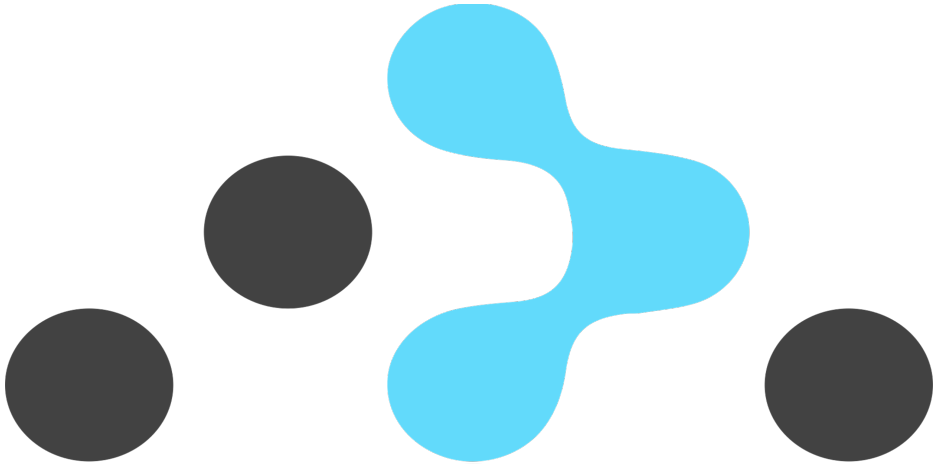


# Navigation

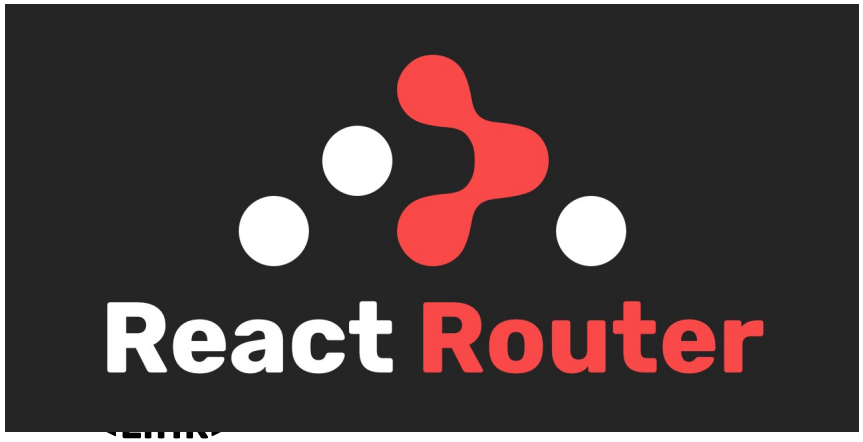
The React Router library

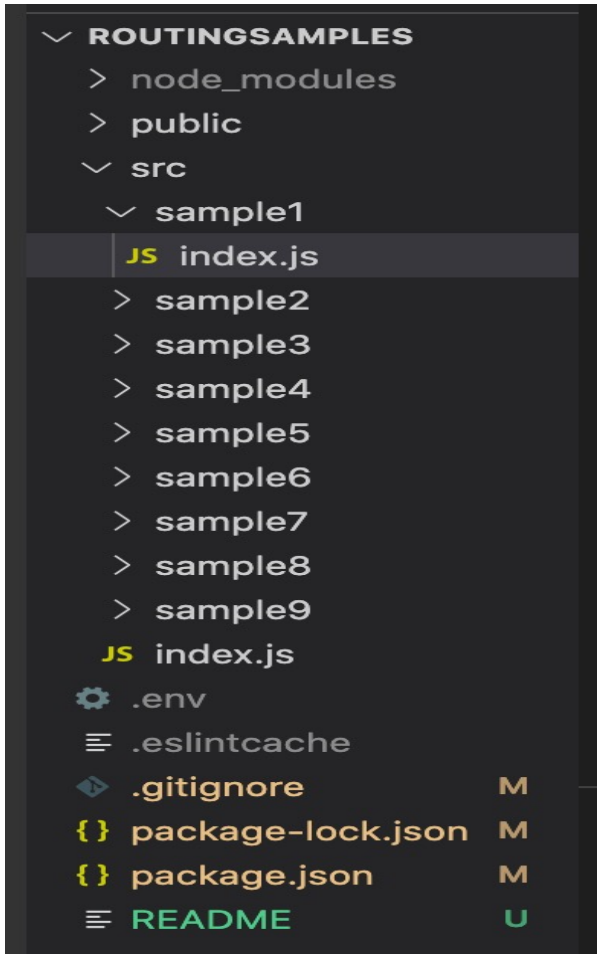
# Routing - Introduction

- **Allows multiple views / pages in an app.**
- **Keeps the URL in sync with the UI.**
- **Adheres to traditional web principles:**
  1. **Addressability.**
  2. **Information sharing.**
  3. **Deep linking.**
  - **1st generation AJAX apps violated these principles.**
- **Not supported by the React framework.**
  - **A separate library is required: React Router.**

# Demos

- **See the archive.**
- **Each sample demos a routing feature**



# Basic routing configuration

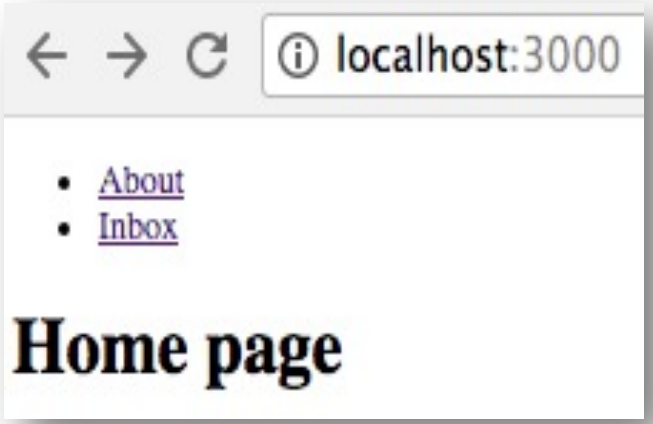
| | URL | Components |
|---|---|---|
| 1 | / | Home |
| 2 | /about | About |
| 3 | /inbox | Inbox |

```
17  const App = () => {
18    return (
19      <BrowserRouter>
20        <Routes>
21          <Route path="/about" element={<About />} />
22          <Route path="/inbox" element={<Inbox />} />
23          <Route index element={<Home />} />
24          <Route path="*" element={<Navigate to="/" replace />} />
25        </Routes>
26      </BrowserRouter>
27    );
28  };
```

- **Declarative routing.**
- <BrowserRouter> **- matches browser's URL with <Route> paths.**
- <Route> **-** element **- what is mounted on DOM when match a occurs.**
  - element **can take any arbitrary JSX.**
  - **Use** index **for root path case (/).**
  - **Use * path for 404 case.**
  - <Navigate> **changes browser's URL address.**
- **App component termed the** Router **– also attached to DOM**
- **Ref. src/sample1**

# Hyperlinks

- **Use the <Link> component for internal links.**
  - **Use anchor tag for external links - <a href . . . . . >**
- **Ref. src/sample2/**




```
6   const Home = () => {
7     return (
8       <>
9         <ul>
10          <li>
11            <Link to="/about">About</Link>
12          </li>
13          <li>
14            <Link to="/inbox">Inbox</Link>
15          </li>
16        </ul>
17        <h1>Home page</h1>
18      </>
19    );
20  };
```

- **<Link> changes browser's URL address (event)**

  **→ React Router handles event by consulting its routing configuration**

  **→ Selected Route's** elements **mounted on DOM → Browser updates screen.**

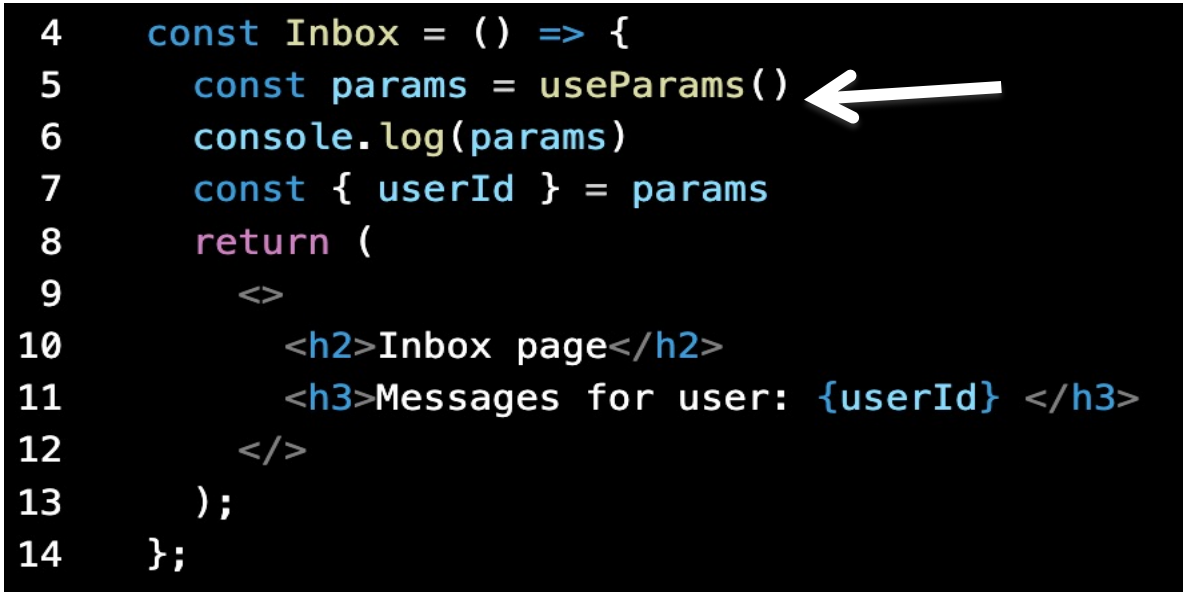
# Dynamic segments.

- **Parameterized URLs, e.g. /users/*22*, /users/*12*/purchases**
  - **How do we declare a parameterized path in the routing configuration?**
  - **How does a component access the parameter value?**
- **Ex: Suppose the Inbox component shows messages for a specific user, where the user's id is part of the browser URL**

  **e.g /inbox/123, where 123 is the user's id.**

- **Solution:** <Route path='/inbox/:userId' element={ <Inbox/> } />
  - **The colon (:) prefixes a parameter in the path;**
  - **Parameter name is arbitrary.**
  - **Ref src/sample3**

# Dynamic segments.



```
4   const Inbox = () => {
5     const params = useParams()
6     console.log(params)
7     const { userId } = params
8     return (
9       <>
10        <h2>Inbox page</h2>
11        <h3>Messages for user: {userId} </h3>
12      </>
13    );
14  };
```

- useParams **hook is provided by React Router library.**
  - **Descructure its object to access parameter value.**
  - **Other useful hooks also provided (see later)**
- **More than one parameter allowed.**

  **e.g. /users/:userId/categories/:categoryName**

# Nested Routes

- Objective**: A component's child is dynamically determined from the browser's URL (Addressability).**

- **EX.: (See src/sample4) Given the route:**

  <Route path='/inbox/:userId' element={ <Inbox /> } />,

  **use the following rules to determine a nested component hierarchy:**

  /inbox/XXX/statistics

  **<Inbox>**

  **<Stats/>**

  **</Inbox>**

  **/inbox/XXX/draft**

  **<Inbox>**

  **<Drafts/>**

  **</Inbox>**

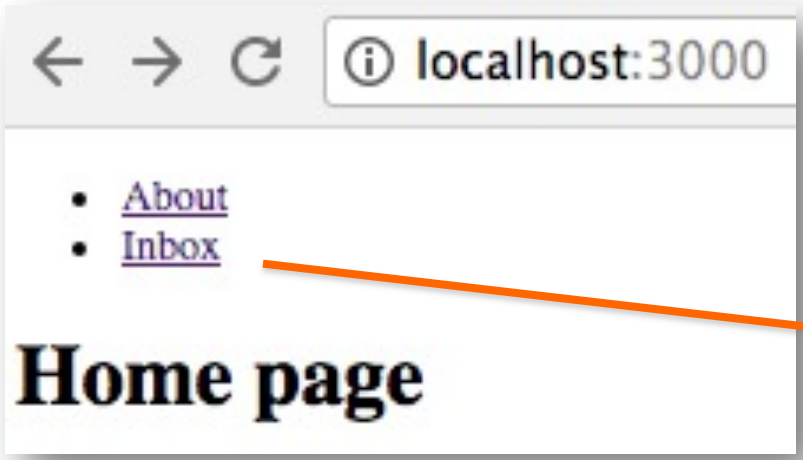
# Nested Routes

```
<BrowserRouter>
  <Routes>
    <Route path="/about" element={<About />} />
    <Route path="/inbox/:userId" element={<Inbox />}>
      <Route path={`statistics`} element={<Stats />} />
      <Route path={`drafts`} element={<Draft />} />
      <Route index element={<Filler />} />
    </Route>
    <Route index element={<Home />} />
    <Route path="*" element={<Navigate to="/" replace />} />
  </Routes>
</BrowserRouter>
```

- **Use RELATIVE path strings for nested <Route> entries.**
- **The index <Route> is optional.**
  - **The default case.**
  - **Avoides ‘blank’ section on screen.**

# Extended <Link>

- **Objective: Pass additional data in a <Link>.**
- **EX.: See /src/sample5/.**



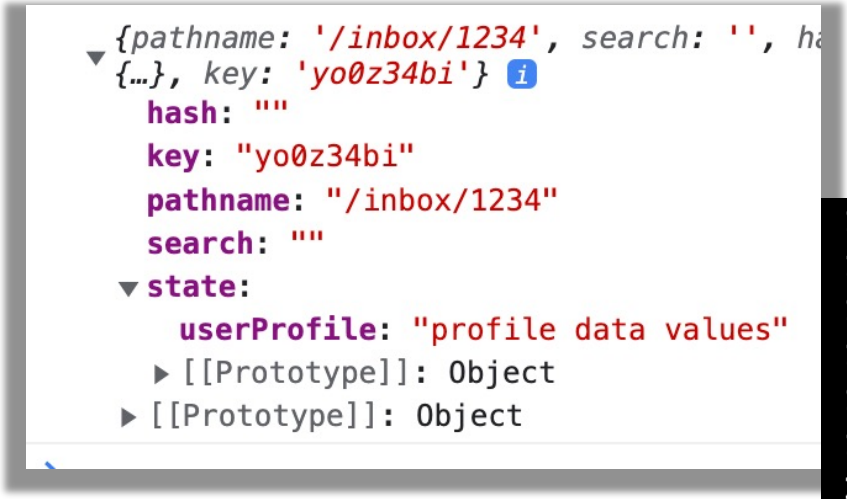

```
31    const userProfile = "profile data values";
32    return (
33      <>
34        <ul>
35          <li>
36            <Link to="/about">About</Link>
37          </li>
38          <li>
39            <Link
40              to={`/inbox/1234`}
41              state={{
42                userProfile:  userProfile,
43              }}
44            >
45              Inbox<span> (Link with extra props
46            </Link>
47          </li>
48        </ul>
```

```
<Route path="/inbox/:userId" element={<Inbox />}
```

- **How does Inbox access the** userProfile **data included in the hyperlink?**
  - **A.: The** useLocation **hook**

# Extended <Link>

- **React Router creates a** location **object each time the URL changes.**




```
14   const Inbox = (props) => {
15     const {userId} = useParams()
16     const locatio = useLocation();
17     console.log(locatio);
18     const {
19       state: { userProfile },
20     } = locatio;
21     return (
22       <>
23         <h2>Inbox page</h2>
24         <p>{`User Id: ${userId}`}</p>
25         <p>{`User profile: ${userProfile}`}</p>
26       </>
27     );
28   };
```

# Routing

- **More later**