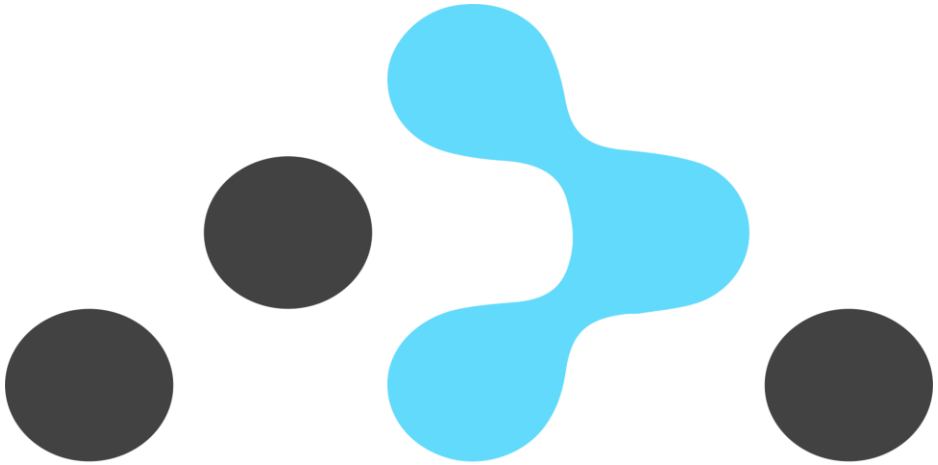


Navigation

The React Router library

Routing - Introduction

- **Allows multiple views in an app.**
  - **But there's only one page (.html) → Single Page App (SPA)**

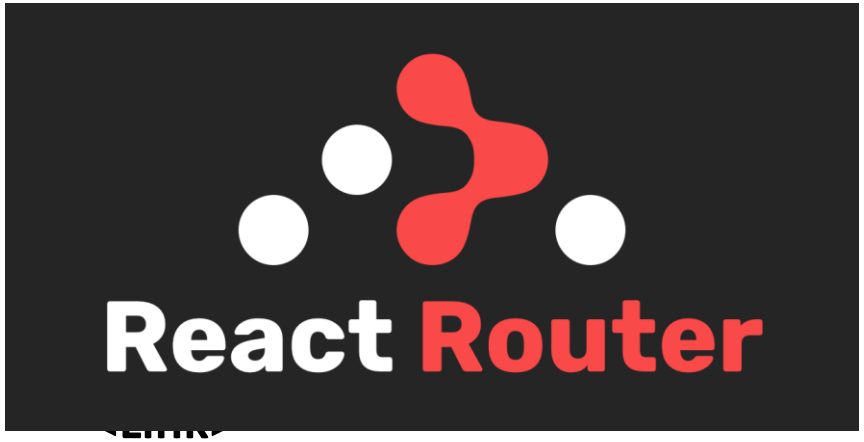
- **Keeps the browser's URL in sync with the UI.**

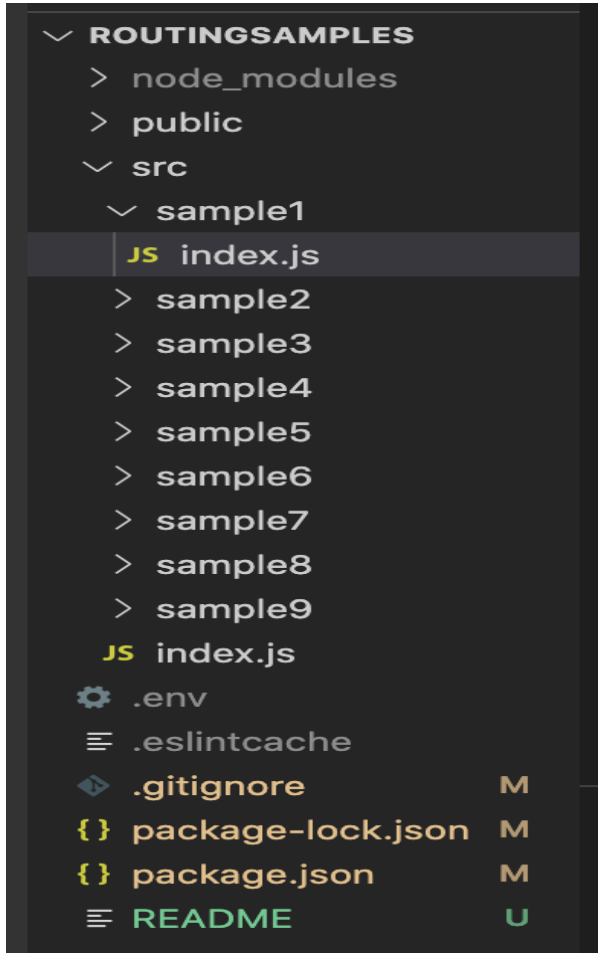
- **Adheres to traditional web principles:**
  1. **Addressability.**
  2. **Information sharing.**
  3. **Deep linking.**

- **Not supported by the React framework.**
  - **A separate library is required: React Router.**

Demos

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





Basic routing configuration

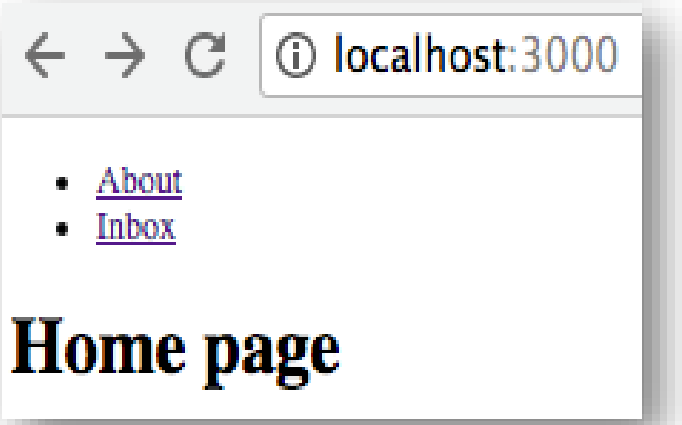
	URL	Components
1	/	Home
2	/about	About
3	/inbox	Inbox

```
const App: React.FC = () => {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/about" element={<About />} />
        <Route path="/inbox" element={<Inbox />} />
        <Route index element={<Home />} />
        <Route path="*" element={<Navigate to="/" replace />} />
      </Routes>
    </BrowserRouter>
  );
```

- **Declarative style.**
- <BrowserRouter> **- matches browser's URL to a <Route>** path**.**
- **Matching Route's** element **is mounted on the DOM.**
  - element **can take any arbitrary TSX.**
  - **Use** index **for root path case (/).**
  - **Use * path for 404 case.**
  - <Navigate> **changes browser's URL address.**
- **App component termed the** Router **component.**
- **Ref. src/sample1**

Hyperlinks

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




```
const Home: React.FC = () => {
  return (
    <>
      <ul>
        <li>
          <Link to="/">Home</Link>
        </li>
        <li>
          <Link to="/about">About</Link>
        </li>
        <li>
          <Link to="/inbox">Inbox</Link>
        </li>
      </ul>
      <h1>Home page</h1>
    </>
  );
};
```

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

  → React Router handles event by consulting its routing configuration

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

Dynamic segments.

- **Parameterised URLs, e.g. /users/*22*, /users/*12*/purchases**
  - **How to declare a parameterised 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.

```
const Inbox:React.FC = () => {
  const params = useParams()
  console.log(params)
  const { userId } = params
  return (
    <>
      <h2>Inbox page</h2>
      <h3>Messages for user: {userId} </h3>
    </>
  );
};
```

- useParams **hook (React Router library).**
  - **Returns an object containing the 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/XYZ/statistics	**/inbox/XYZ/draft**
<Inbox>	**<Inbox>**
<Stats/>	**<Drafts/>**
</Inbox>	**</Inbox>**

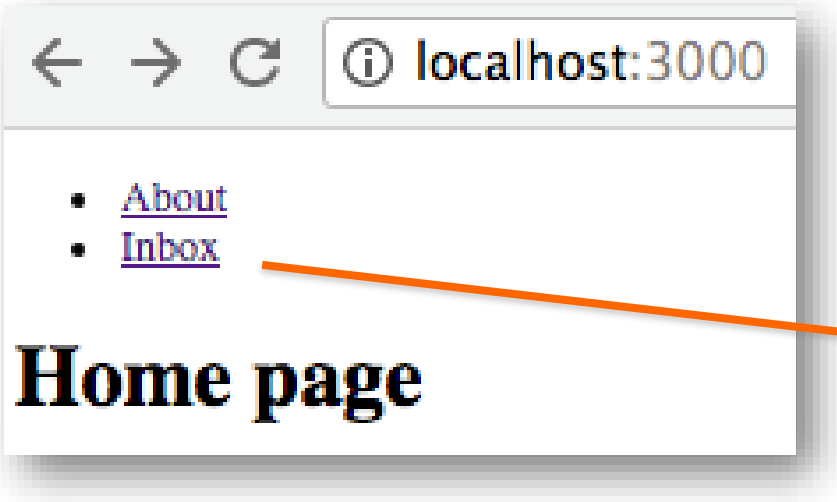
Nested Routes

```
const App = () => {
  return (
    <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 in the nested <Route> entries.**
- **The index <Route> is optional.**
  - **For the default case.**
  - **Avoids a ‘blank’ section on screen.**
- **Use <Outlet/> as a placeholder in the container component**

Extended <Link>

- **Objective: Supply data to the component mounted by 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.**

```
{pathname: '/inbox/1234', search: '', ha
{…}, key: 'yo0z34bi'}
  hash: ""
  key: "yo0z34bi"
  pathname: "/inbox/1234"
  search: ""
  state:
    userProfile: "profile data values"
    [[Prototype]]: Object
  [[Prototype]]: Object
```

```
const Inbox = () => {
  const { userId } = useParams()
  const location = useLocation();
  console.log(location);
  const {
    state: { userProfile },
  } = location;
  return (
    <>
      <h2>Inbox page</h2>
      <p>{`User Id: ${userId}`}</p>
      <p>{`User profile: ${userProfile}`}</p>
    </>
  );
};
```

Routing

- **More later**