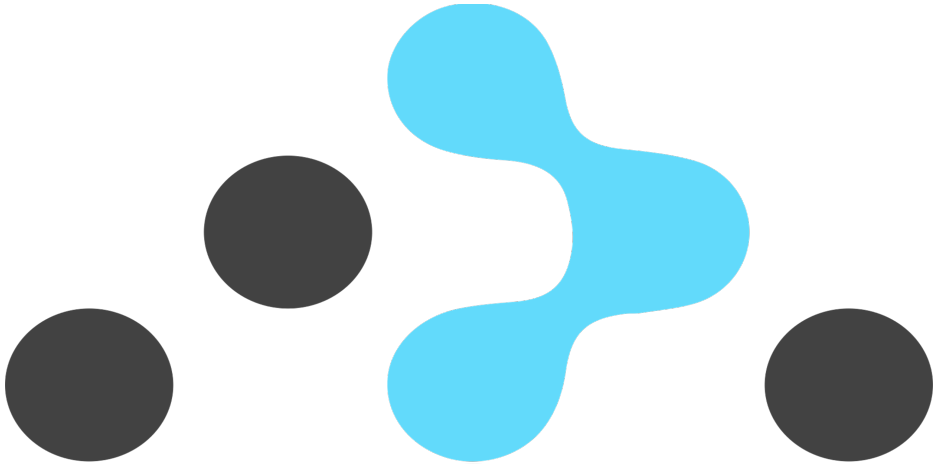


Navigation

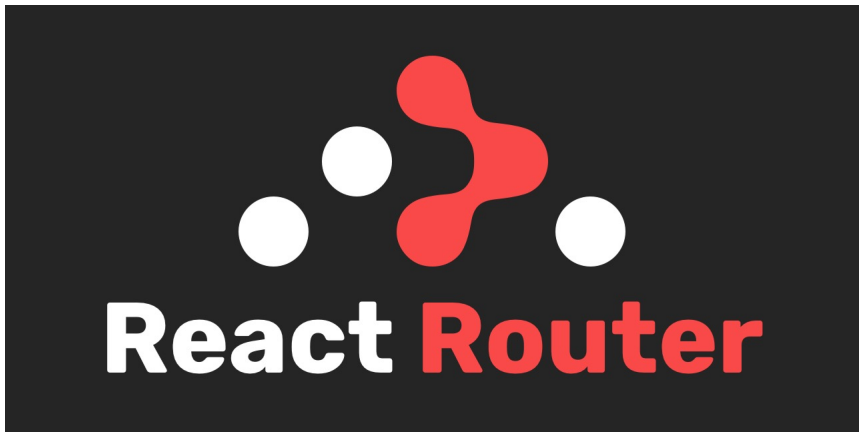
The React Router

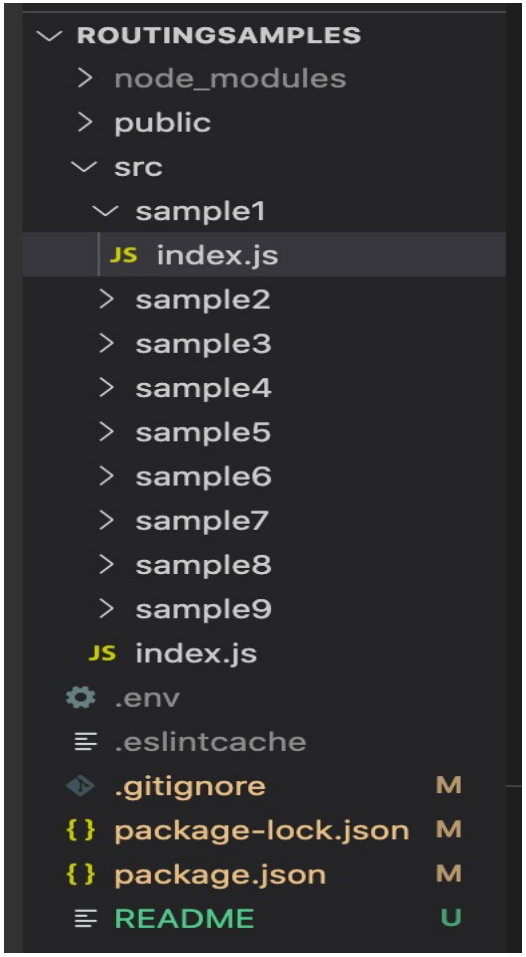
Routing - Introduction

- **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

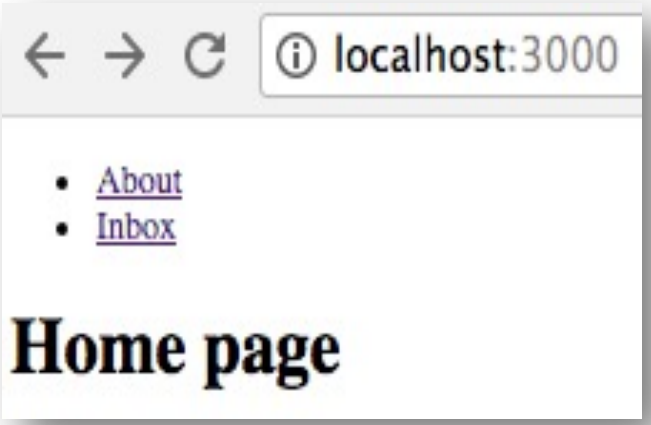
	URL	Components
1	/	Home
2	/about	About
3	/inbox	Inbox

```
17  const App = () => {
18    return (
19      <BrowserRouter>
20        <Switch>
21          <Route path="/about" component={About} />
22          <Route path="/inbox" component={Inbox} />
23          <Route exact path="/" component={Home} />
24          <Redirect from="*" to="/" />
25        </Switch>
26      </BrowserRouter>
27    );
28  };
29
30  ReactDOM.render(<App />, document.getElementById("root"));
```

- **Declarative routing.**
- **<BrowserRouter> - matchs browser's URL with a <Route> path.**
- **Matched <Route> declares component to be mounted.**
- **<Route> path supports regular expression pattern matching.**
  - **Use exact prop for precision.**
- **Use <Redirect> to avoid HTTP 404 error.**
- **<Switch> - only one of the nested Routs can be active.**
- **ReactDOM.render() passed an app's Router component.**
- **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  };
```

Absolute URL

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

  → React Router handles event by consulting its routing configuration

  → Component unmounting/mounting occurs → Browser updates screen

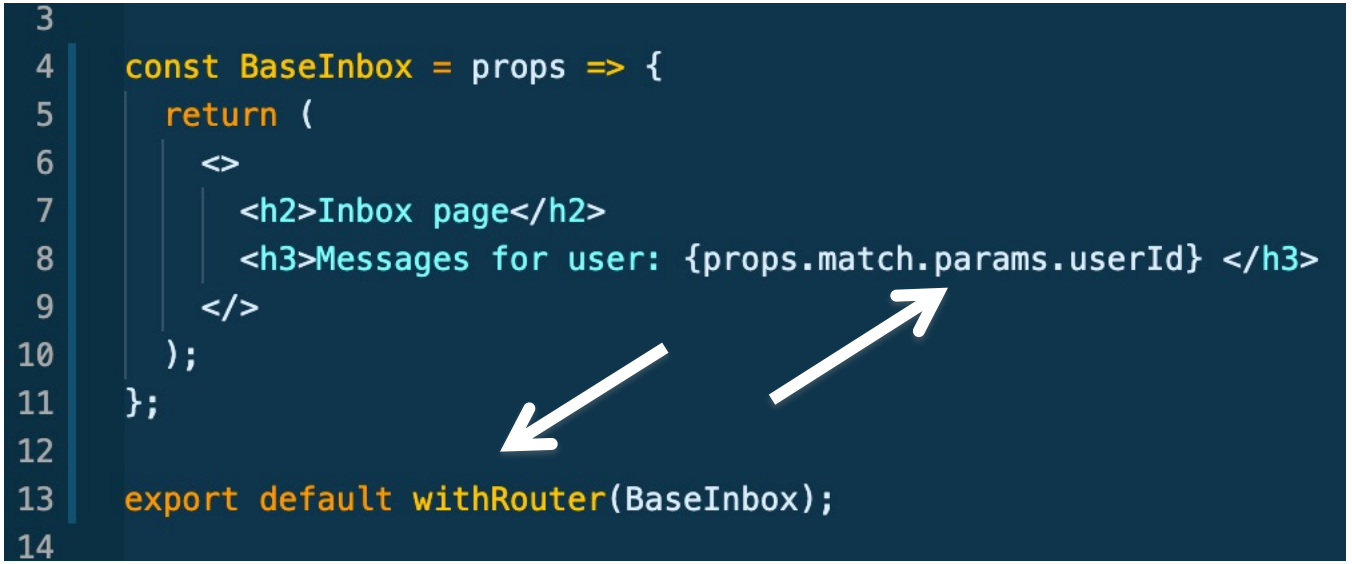
Dynamic segments.

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

Dynamic segments.



```
3
4   const BaseInbox = props => {
5     return (
6       <>
7         <h2>Inbox page</h2>
8         <h3>Messages for user: {props.match.params.userId} </h3>
9       </>
10    );
11  };
12
13  export default withRouter(BaseInbox);
14
```

- **withRouter() function: Returns a new, enriched component.**
  - **Injects routing props into a component:**
    - **props.match.params.(parameter-name)**
    - **props.history**
- **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' component={ Inbox } />,

  when the browser URL is:

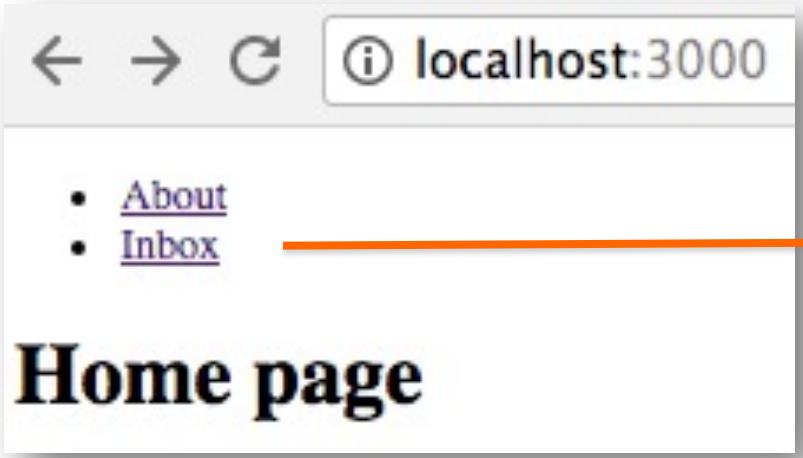
  1. **/inbox/XXX/statistics render Inbox + Stats components.**
  2. **/inbox/XXX/draft then render Inbox + Drafts**

```
const BaseInbox = (props) => {
  return (
    <>
      <h1>Inbox page</h1>
      <Messages id={props.match.params.userId} />
      <Route path={`/inbox/:userId/statistics`} component={Stats} />
      <Route path={`/inbox/:userId/drafts`} component={Draft} />
    </>
  );
};
```

Nested routes

Extended <Link>

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




```
const userId = 'id1234'
const beta = 'something else'

<Link
    to={{
    pathname: "/inbox",
    state: {
        user: userId,
        beta: beta
    }
    }}
>Inbox </Link>
```

```
const Inbox = props => {
  const { user, beta } = props.location.state;
  return (
    <>
      <h2>Inbox page</h2>
      <p>{`Link Props: ${user}, ${beta}`}</p>
    </>
  );
};
```

```
<Route path="/inbox" component={Inbox} />
```

Routing

- **More later**