

Navigation

The React Router

Introduction

- **Allows** multiple views **and** flows **in an app**.
- **Keeps the URL in sync with the UI**.
- **Supports traditional web principles:**
 - 1. Addressability.**
 - 2. Information sharing.**
 - 3. Deep linking.**
 - **1st generation AJAX apps violated these principles.**
- **A separate library.**

Basic routing configuration

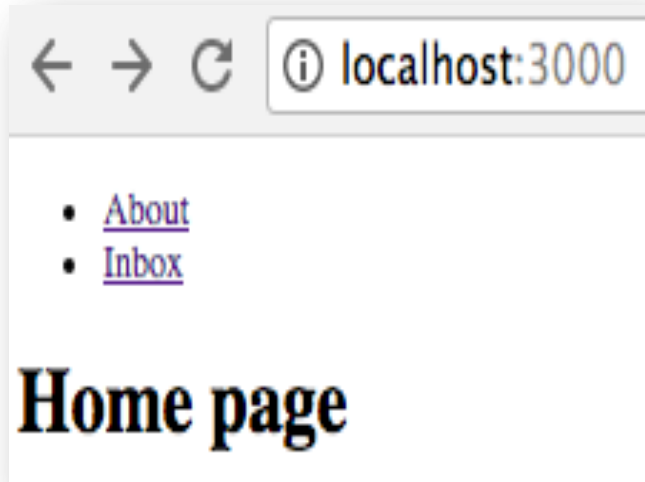
	URL	Components
1	/	Home
2	/about	About
3	/inbox	Inbox

```
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"));
31
```

- **Declarative routing.**
- **<BrowserRouter> match browser's URL with <Route> paths.**
- **<Route> component mounted when match occurs.**
- **<Route> path supports regular expression pattern matching.**
 - **Use exact argument for precision.**
- **Use <Redirect> to avoid 404-type 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 >`
- EX. 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>` gives access to other useful router properties.
- Use `<LinkContainer>` when link wraps other 3rd party component, e.g. Bootstrap-React `<Button />`

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 custom component access the parameter value?

- **Ex: Ref** src/sample3/.


Suppose the Inbox component shows messages for a specific user, based on the browser URL e.g /inbox/123 where 123 is the user's id.

```
<Route path='/inbox/:userId' component={ Inbox } />
```

**The colon (:) prefixes a parameter in the path.
Parameter name is arbitrary.**

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
```



The diagram consists of two white arrows. One arrow originates from the `withRouter` function call in line 13 and points towards the `BaseInbox` component definition in line 4. The second arrow originates from the `props` parameter in the `BaseInbox` definition and points towards the JSX elements in lines 6-9, specifically highlighting the dynamic segment `{props.match.params.userId}` in line 8.

- **withRouter() function:**
 - **Injects routing props into a component:**
 - `props.match.params.(parameter-name)`
 - `props.history`
 - **Returns a new, enriched component.**

Routing.

- **More routing features next week.**

