# 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.**
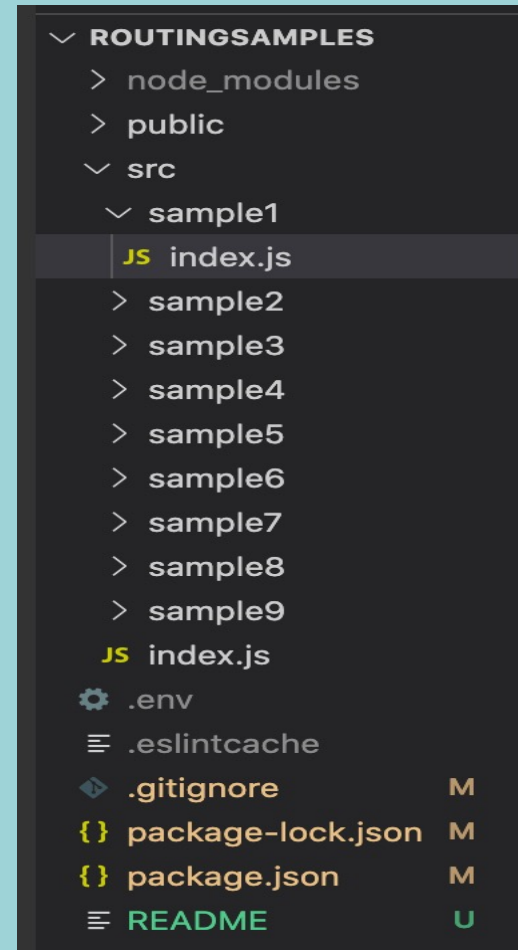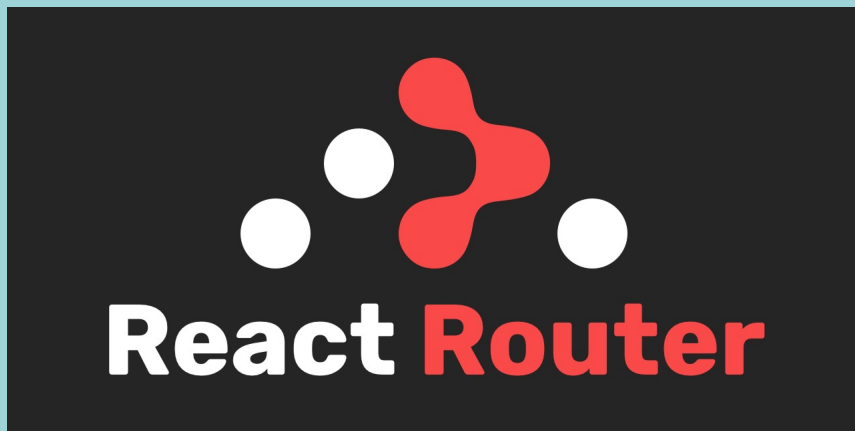
Basic routing configuration

	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"));
31
```

- **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 Routes can be active.**
- **ReactDOM.render() passed an app's Router component.**
- **Ref. src/sample1**

4

Hyperlinks

- **Use the \<Link\> component for internal links.**
  - **Use anchor tag for external links - \<a href . . . . . \>**
- **Ref. src/sample2/**
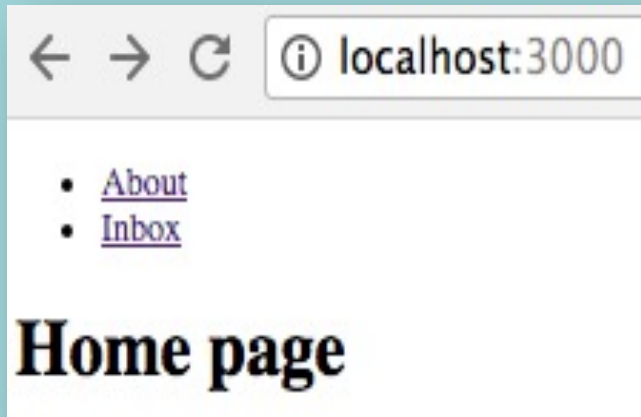
```
 6    const Home = () => {
 7      return (
 8        <>
 9          <ul>
10            <li>
11              <Link to="/about">About</Link>
12            </li>
13            <li>
14              <Link to="/inbox">Inbox</Link>
15            </li>
16          </ul>
17          <h1>Home page</h1>
18        </>
19      );
20    };
```
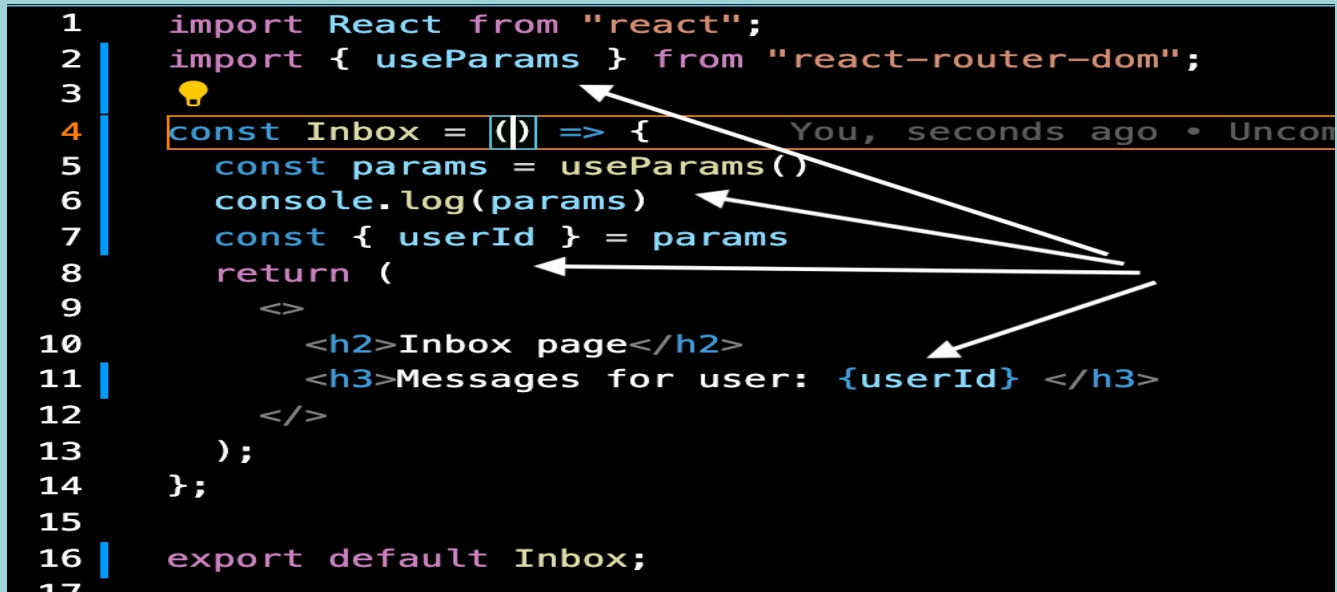
Absolute URL

localhost:3000

- About
- Inbox

Home page

- **\<Link\> changes browser's URL address (event)**
  → 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 the mounted 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 } />**
  - **Colon (:) prefixes a parameter in the path; Parameter name is arbitrary.**
  - **Ref src/sample3**

Dynamic segments.

```
1    import React from "react";
2    import { useParams } from "react-router-dom";
3
4    const Inbox = (()) => {              You, seconds ago • Uncon
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   };
15
16   export default Inbox;
17
```

- **The** useParams **hook.**

- **More than one parameter allowed.**
 - **e.g. /users/:userId/categories/:categoryName**

7

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**

```
 4   const Inbox = (props) => {
 5     const { userId } = useParams()        You, seconds ago • Uncommitted
 6     return (
 7       <>
 8         <h1>Inbox page</h1>
 9         <Messages id={userId} />
10         <Route path={`/inbox/:uid/statistics`} component={Stats} />
11         <Route path={`/inbox/:uid/drafts`} component={Draft} />
12       </>
13     );
14   };
```

Nested routes

Extended <Link>

- **Objective: Pass additional props via a <Link>.**

- **EX.: See /src/sample5/.**
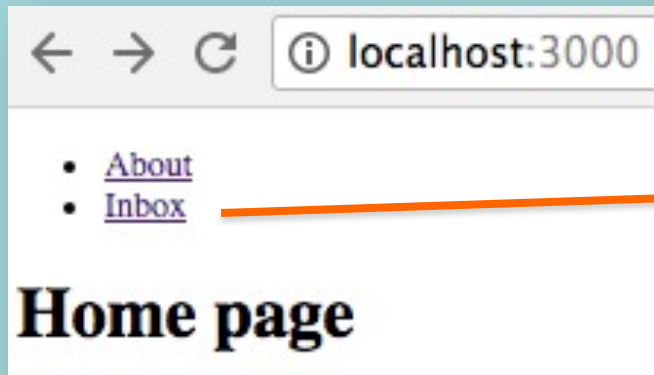
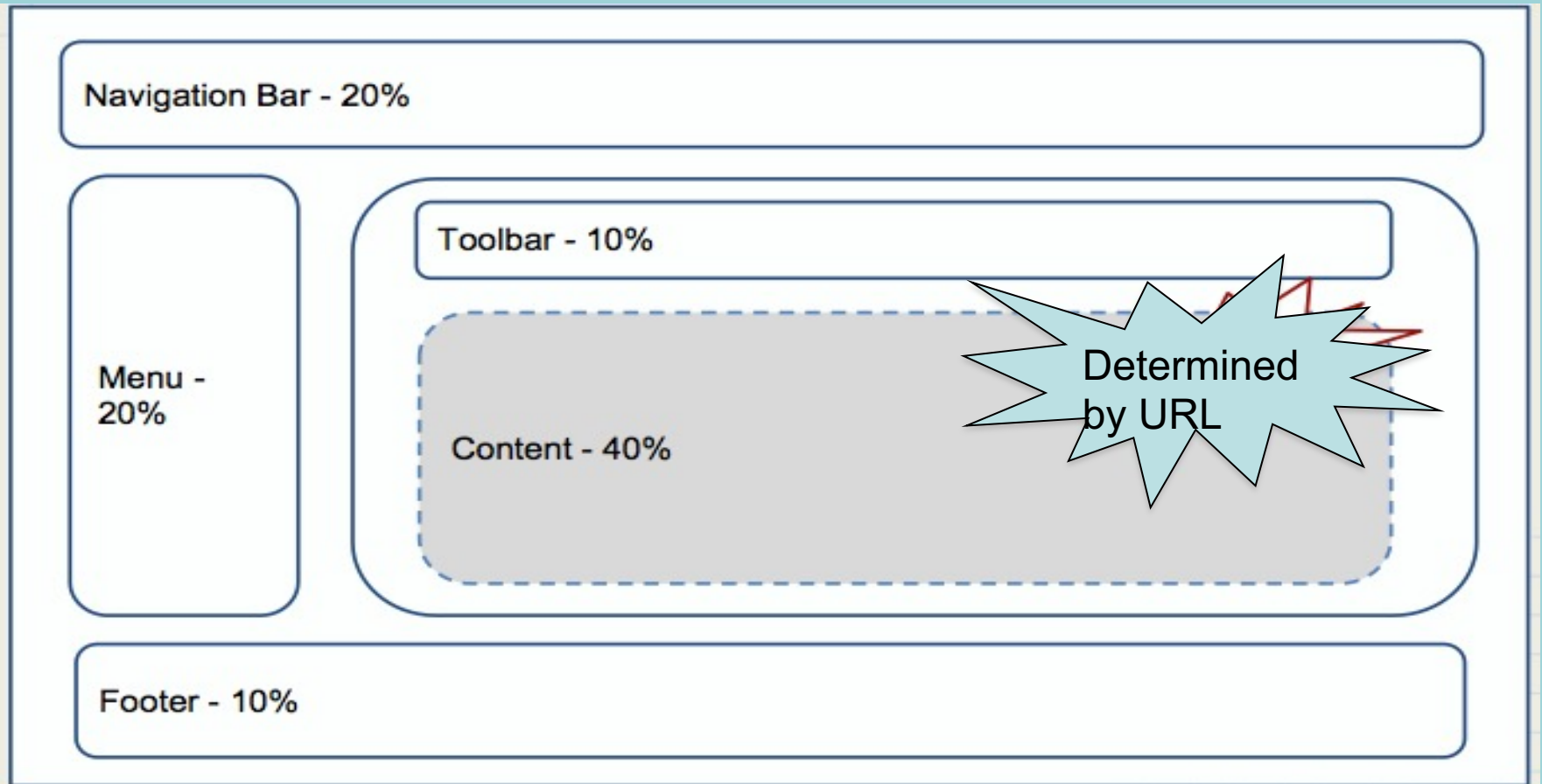

```
2    const userId = 'id1234'
3    const beta = 'something else'
4
5    <Link
6        to={{
7        pathname: "/inbox",
8        state: {
9            user: userId,
10           beta: beta
11       }
12       }}
13   >Inbox </Link>
14
```

```
6    const Inbox = props => {
7      const locatio = useLocation();
8      console.log(locatio)
9      const { state: { user, beta } } = locatio;        Y
10     return (
11       <>
12         <h2>Inbox page</h2>
13         <p>{`Link Props: ${user}, ${beta}`}</p>
14       </>
15     );
16   };
```
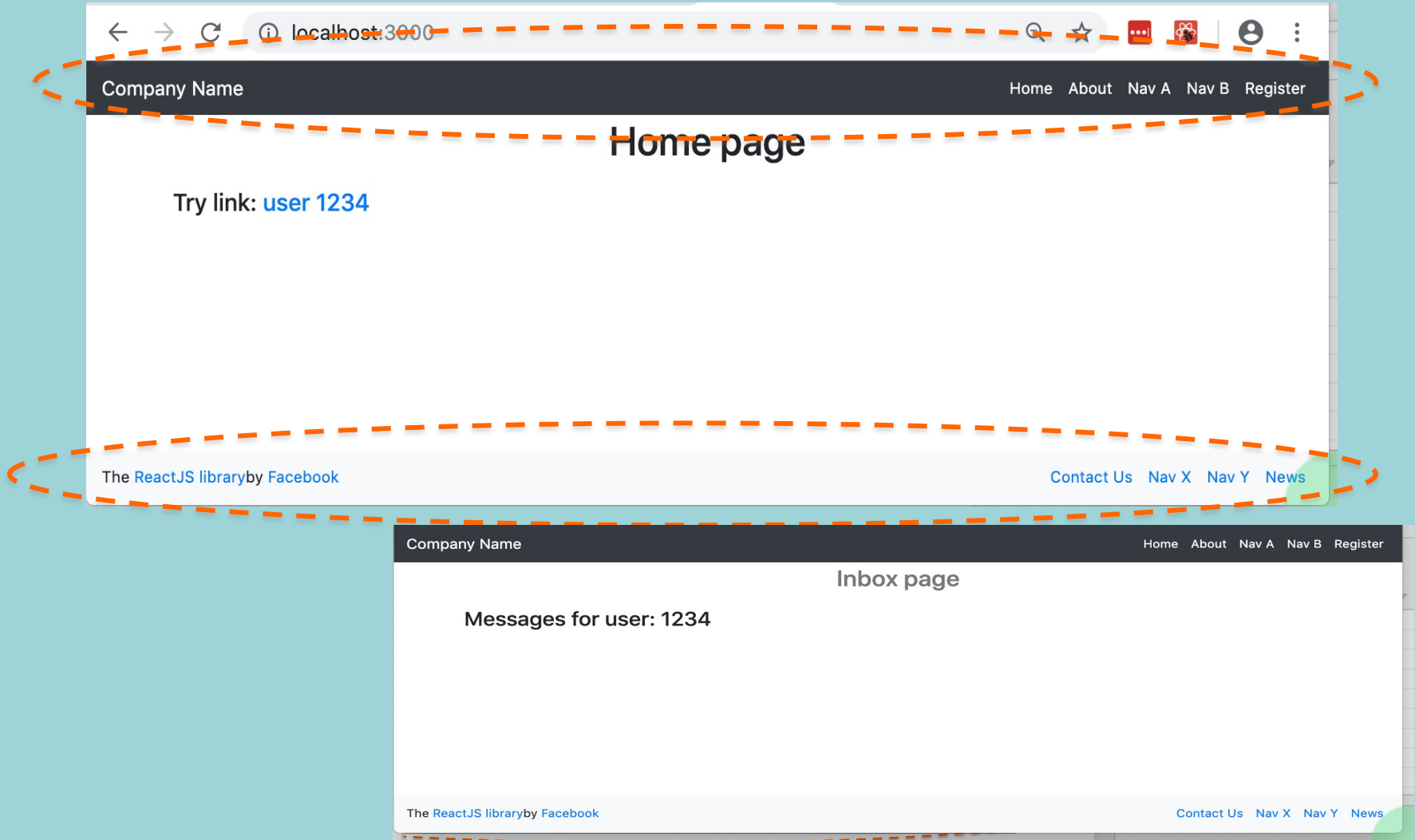
```
<Route path="/inbox" component={Inbox} />
```

Typical Web app layout

Navigation Bar - 20%

Menu - 20%

Toolbar - 10%

Content - 40%

Determined by URL

Footer - 10%

Persistent elements/components

- **Use cases: Site-wide Headers, Footers, Side menus.**

Persistent elements/components

- **Ref.** src/sample6

```
class Router extends Component {
    render() {
        return (
            <BrowserRouter>
                <div>
                    <Header/>
                    <div className="container">
                        <Switch>
                            <Route path='/about' component={ About } />
                            <Route path='/register' component={ Register } />
                            <Route path='/contact' component={ Contact } />
                            <Route path='/inbox/:userId' component={ Inbox } />
                            <Route exact path='/' component={ Home } />
                            <Redirect from='*' to='/' />
                        </Switch>
                    </div>
                    <Footer />
                </div>
            </BrowserRouter>
        )
    }
}
```

Alternative <Route> API.

- **To-date:** *<Route path={…URL path…} component={ ComponentX} />*
  - **Mounted component always gets a default prop object.**
- **Disadv.: We cannot pass custom props to the mounted component.**

- **Alternative:**

  <Route path={…URL path…} render={…function….}>
  – **where** *function* **return the mounted component.**
- **EX.: See** /src/sample7/**.**

  Objective: Pass usage data to the <Stats> **component from** sample4's nested Route.

```
<Route path={`/inbox/:userId/statistics`} component={Stats} />
```

Alternative <Route> API.

```
<Route
  path={`/inbox/:id/statistics`}
  render={ (props) => {
    return <Stats {...props} usage={[5.4, 9.2]} />;
  }}
/>
```

- **The** render **prop function argument is the inherited props object.**

```
const Stats = (props) => {
  return (
    <>
      <h3>Statistical data for user: {props.match.params.id}</h3>
      <h4>Emails sent (per day) = {props.usage[0]} </h4>
      <h4>Emails received (per day) = {props.usage[1]} </h4>
    </>
  );
};
```

Routing

- **More later**