React Router logo

REMIX / REACT ROUTER

CORE

WEB

NATIVE

ANNOUNCEMENTS

The Future of React Router

EXAMPLES

Basic

URL Parameters

Nesting

Redirects (Auth)

Custom Link

Preventing Transitions

No Match (404)

Recursive Paths

Sidebar

Animated Transitions

Route Config

Modal Gallery

StaticRouter Context

Query Parameters

GUIDES

Quick Start

Primary Components

Server Rendering

Code Splitting

Scroll Restoration

Philosophy

Testing

Deep Redux Integration

Static Routes

API

Hooks

useHistory

useLocation

useParams

useRouteMatch

<BrowserRouter>

basename: string

getUserConfirmation: func

forceRefresh: bool

keyLength: number

children: node

<HashRouter>

basename: string

getUserConfirmation: func

hashType: string

children: node

<Link>

to: string

to: object

to: func

replace: bool

innerRef: func

innerRef: RefObject

component: React.Component

others

\<NavLink>

className: string | func

activeClassName: string

style: object | func

activeStyle: object

exact: bool

strict: bool

isActive: func

location: object

aria-current: string

\<Prompt>

\<MemoryRouter>

initialEntries: array

initialIndex: number

getUserConfirmation: func

keyLength: number

children: node

\<Redirect>

to: string

to: object

push: bool

from: string

exact: bool

strict: bool

sensitive: bool

\<Route>

Route render methods

Route props

component

RENDER: FUNC
CHILDREN: FUNC
PATH: STRING | STRING[]
EXACT: BOOL
STRICT: BOOL
LOCATION: OBJECT
SENSITIVE: BOOL
<Router>
HISTORY: OBJECT
CHILDREN: NODE
<StaticRouter>
BASENAME: STRING
LOCATION: STRING
LOCATION: OBJECT
CONTEXT: OBJECT
CHILDREN: NODE
<Switch>
LOCATION: OBJECT
CHILDREN: NODE
generatePath
PATTERN: STRING
PARAMS: OBJECT
HISTORY
HISTORY IS MUTABLE
LOCATION
MATCH
NULL MATCHES
matchPath
PATHNAME

PROPS

RETURNS

withRouter

Component.WrappedComponent

wrappedComponentRef: func

Quick Start

To get started with React Router in a web app, you'll need a React web app. If you need to create one, we recommend you try Create React App. It's a popular tool that works really well with React Router.First, install create-react-app and make a new project with it.npx create-react-app demo-app

cd demo-app

Installation

You can install React Router from the public npm registry with either npm or yarn. Since we're building a web app, we'll use react-router-dom in this guide.yarn add react-router-dom

Next, copy/paste either of the following examples into src/App.js.

1st Example: Basic Routing

In this example we have 3 "pages" handled by the router: a home page, an about page, and a users page. As you click around on the different <Link>s, the router renders the matching <Route>.Note: Behind the

scenes a <Link> renders an <a> with a real href, so people using the keyboard for navigation or screen readers will still be able to use this app.import React from "react";
import {
  BrowserRouter as Router,
  Switch,
  Route,
  Link
} from "react-router-dom";

export default function App() {
  return (
    <Router>
      <div>
        <nav>

              <Link to="/">Home</Link>

              <Link to="/about">About</Link>

              <Link to="/users">Users</Link>

        </nav>

```jsx
      {/* A <Switch> looks through its children <Route>s and
          renders the first one that matches the current URL. */}
      <Switch>
        <Route path="/about">
          <About />
        </Route>
        <Route path="/users">
          <Users />
        </Route>
        <Route path="/">
          <Home />
        </Route>
      </Switch>
    </div>
  </Router>
  );
}

function Home() {
  return <h2>Home</h2>;
}

function About() {
  return <h2>About</h2>;
}
```

```
function Users() {
  return <h2>Users</h2>;
}
```

2nd Example: Nested Routing

This example shows how nested routing works. The route /topics loads the Topics component, which renders any further <Route>'s conditionally on the paths :id value.

```
import React from "react";
import {
  BrowserRouter as Router,
  Switch,
  Route,
  Link,
  useRouteMatch,
  useParams
} from "react-router-dom";

export default function App() {
  return (
    <Router>
      <div>
        <ul>
          <li>
            <Link to="/">Home</Link>
          </li>
          <li>
            <Link to="/about">About</Link>
          </li>
```

```
      <li>
        <Link to="/topics">Topics</Link>
      </li>
    </ul>

    <Switch>
      <Route path="/about">
        <About />
      </Route>
      <Route path="/topics">
        <Topics />
      </Route>
      <Route path="/">
        <Home />
      </Route>
    </Switch>
  </div>
 </Router>
);
}

function Home() {
 return <h2>Home</h2>;
}

function About() {
 return <h2>About</h2>;
}
```

```
function Topics() {
  let match = useRouteMatch();

  return (
    <div>
      <h2>Topics</h2>

      <ul>
        <li>
          <Link to={`${match.url}/components`}>Components</Link>
        </li>
        <li>
          <Link to={`${
```