[**REACT TRAINING**](https://reacttraining.com/)

**/**

[**REACT ROUTER**](https://github.com/ReactTraining/react-router)

[**WEB**](https://reacttraining.com/react-router/web)[**NATIVE**](https://reacttraining.com/react-router/native)[**CORE**](https://reacttraining.com/react-router/core)

**EXAMPLES**

[Basic](https://reacttraining.com/react-router/web/example/basic)[URL Parameters](https://reacttraining.com/react-router/web/example/url-params)[Redirects (Auth)](https://reacttraining.com/react-router/web/example/auth-workflow)[Custom Link](https://reacttraining.com/react-router/web/example/custom-link)[Preventing Transitions](https://reacttraining.com/react-router/web/example/preventing-transitions)[No Match (404)](https://reacttraining.com/react-router/web/example/no-match)[Recursive Paths](https://reacttraining.com/react-router/web/example/recursive-paths)[Sidebar](https://reacttraining.com/react-router/web/example/sidebar)[Animated Transitions](https://reacttraining.com/react-router/web/example/animated-transitions)[Ambiguous Matches](https://reacttraining.com/react-router/web/example/ambiguous-matches)[Route Config](https://reacttraining.com/react-router/web/example/route-config)[Modal Gallery](https://reacttraining.com/react-router/web/example/modal-gallery)[StaticRouter Context](https://reacttraining.com/react-router/web/example/static-router)

**GUIDES**

[Philosophy](https://reacttraining.com/react-router/web/guides/philosophy)[Basic Components](https://reacttraining.com/react-router/web/guides/basic-components)[Quick Start](https://reacttraining.com/react-router/web/guides/quick-start)[Server Rendering](https://reacttraining.com/react-router/web/guides/server-rendering)[Code Splitting](https://reacttraining.com/react-router/web/guides/code-splitting)[Scroll Restoration](https://reacttraining.com/react-router/web/guides/scroll-restoration)[Testing](https://reacttraining.com/react-router/web/guides/testing)[Redux Integration](https://reacttraining.com/react-router/web/guides/redux-integration)[Static Routes](https://reacttraining.com/react-router/web/guides/static-routes)[Dealing with Update Blocking](https://reacttraining.com/react-router/web/guides/dealing-with-update-blocking)

**API**

[<BrowserRouter>](https://reacttraining.com/react-router/web/api/BrowserRouter)

[basename: string](https://reacttraining.com/react-router/web/api/BrowserRouter/basename-string)[getUserConfirmation: func](https://reacttraining.com/react-router/web/api/BrowserRouter/getuserconfirmation-func)[forceRefresh: bool](https://reacttraining.com/react-router/web/api/BrowserRouter/forcerefresh-bool)[keyLength: number](https://reacttraining.com/react-router/web/api/BrowserRouter/keylength-number)[children: node](https://reacttraining.com/react-router/web/api/BrowserRouter/children-node)

[<HashRouter>](https://reacttraining.com/react-router/web/api/HashRouter)

[basename: string](https://reacttraining.com/react-router/web/api/HashRouter/basename-string)[getUserConfirmation: func](https://reacttraining.com/react-router/web/api/HashRouter/getuserconfirmation-func)[hashType: string](https://reacttraining.com/react-router/web/api/HashRouter/hashtype-string)[children: node](https://reacttraining.com/react-router/web/api/HashRouter/children-node)

[<Link>](https://reacttraining.com/react-router/web/api/Link)

[to: string](https://reacttraining.com/react-router/web/api/Link/to-string)[to: object](https://reacttraining.com/react-router/web/api/Link/to-object)[replace: bool](https://reacttraining.com/react-router/web/api/Link/replace-bool)[innerRef: function](https://reacttraining.com/react-router/web/api/Link/innerref-function)[others](https://reacttraining.com/react-router/web/api/Link/others)

[<NavLink>](https://reacttraining.com/react-router/web/api/NavLink)

[activeClassName: string](https://reacttraining.com/react-router/web/api/NavLink/activeclassname-string)[activeStyle: object](https://reacttraining.com/react-router/web/api/NavLink/activestyle-object)[exact: bool](https://reacttraining.com/react-router/web/api/NavLink/exact-bool)[strict: bool](https://reacttraining.com/react-router/web/api/NavLink/strict-bool)[isActive: func](https://reacttraining.com/react-router/web/api/NavLink/isactive-func)[location: object](https://reacttraining.com/react-router/web/api/NavLink/location-object)

[<Prompt>](https://reacttraining.com/react-router/web/api/Prompt)

[<MemoryRouter>](https://reacttraining.com/react-router/web/api/MemoryRouter)

[initialEntries: array](https://reacttraining.com/react-router/web/api/MemoryRouter/initialentries-array)[initialIndex: number](https://reacttraining.com/react-router/web/api/MemoryRouter/initialindex-number)[getUserConfirmation: func](https://reacttraining.com/react-router/web/api/MemoryRouter/getuserconfirmation-func)[keyLength: number](https://reacttraining.com/react-router/web/api/MemoryRouter/keylength-number)[children: node](https://reacttraining.com/react-router/web/api/MemoryRouter/children-node)

[<Redirect>](https://reacttraining.com/react-router/web/api/Redirect)

[to: string](https://reacttraining.com/react-router/web/api/Redirect/to-string)[to: object](https://reacttraining.com/react-router/web/api/Redirect/to-object)[push: bool](https://reacttraining.com/react-router/web/api/Redirect/push-bool)[from: string](https://reacttraining.com/react-router/web/api/Redirect/from-string)[exact: bool](https://reacttraining.com/react-router/web/api/Redirect/exact-bool)[strict: bool](https://reacttraining.com/react-router/web/api/Redirect/strict-bool)

[<Route>](https://reacttraining.com/react-router/web/api/Route)

[Route render methods](https://reacttraining.com/react-router/web/api/Route/route-render-methods)[Route props](https://reacttraining.com/react-router/web/api/Route/route-props)[component](https://reacttraining.com/react-router/web/api/Route/component)[render: func](https://reacttraining.com/react-router/web/api/Route/render-func)[children: func](https://reacttraining.com/react-router/web/api/Route/children-func)[path: string](https://reacttraining.com/react-router/web/api/Route/path-string)[exact: bool](https://reacttraining.com/react-router/web/api/Route/exact-bool)[strict: bool](https://reacttraining.com/react-router/web/api/Route/strict-bool)[location: object](https://reacttraining.com/react-router/web/api/Route/location-object)[sensitive: bool](https://reacttraining.com/react-router/web/api/Route/sensitive-bool)

[<Router>](https://reacttraining.com/react-router/web/api/Router)

[history: object](https://reacttraining.com/react-router/web/api/Router/history-object)[children: node](https://reacttraining.com/react-router/web/api/Router/children-node)

[<StaticRouter>](https://reacttraining.com/react-router/web/api/StaticRouter)

[basename: string](https://reacttraining.com/react-router/web/api/StaticRouter/basename-string)[location: string](https://reacttraining.com/react-router/web/api/StaticRouter/location-string)[location: object](https://reacttraining.com/react-router/web/api/StaticRouter/location-object)[context: object](https://reacttraining.com/react-router/web/api/StaticRouter/context-object)[children: node](https://reacttraining.com/react-router/web/api/StaticRouter/children-node)

[<Switch>](https://reacttraining.com/react-router/web/api/Switch)

[location: object](https://reacttraining.com/react-router/web/api/Switch/location-object)[children: node](https://reacttraining.com/react-router/web/api/Switch/children-node)

[history](https://reacttraining.com/react-router/web/api/history)

[history is mutable](https://reacttraining.com/react-router/web/api/history/history-is-mutable)

[location](https://reacttraining.com/react-router/web/api/location)

[match](https://reacttraining.com/react-router/web/api/match)

[null matches](https://reacttraining.com/react-router/web/api/match/null-matches)

[matchPath](https://reacttraining.com/react-router/web/api/matchPath)

[pathname](https://reacttraining.com/react-router/web/api/matchPath/pathname)[props](https://reacttraining.com/react-router/web/api/matchPath/props)

[withRouter](https://reacttraining.com/react-router/web/api/withRouter)

[Component.WrappedComponent](https://reacttraining.com/react-router/web/api/withRouter/componentwrappedcomponent)[wrappedComponentRef: func](https://reacttraining.com/react-router/web/api/withRouter/wrappedcomponentref-func)

[**<BrowserRouter>**](https://reacttraining.com/web/api/BrowserRouter)

A [<Router>](https://reacttraining.com/core/api/Router) that uses the HTML5 history API (pushState, replaceState and the popstate event) to keep your UI in sync with the URL.

import { BrowserRouter } from 'react-router-dom' <BrowserRouter basename={optionalString} forceRefresh={optionalBool} getUserConfirmation={optionalFunc} keyLength={optionalNumber} > <App/> </BrowserRouter>

[**basename: string**](https://reacttraining.com/web/api/BrowserRouter/basename-string)

The base URL for all locations. If your app is served from a sub-directory on your server, you’ll want to set this to the sub-directory. A properly formatted basename should have a leading slash, but no trailing slash.

<BrowserRouter basename="/calendar"/> <Link to="/today"/> // renders <a href="/calendar/today">

[**getUserConfirmation: func**](https://reacttraining.com/web/api/BrowserRouter/getuserconfirmation-func)

A function to use to confirm navigation. Defaults to using [window.confirm](https://developer.mozilla.org/en-US/docs/Web/API/Window/confirm).

// this is the default behavior const getConfirmation = (message, callback) => { const allowTransition = window.confirm(message) callback(allowTransition) } <BrowserRouter getUserConfirmation={getConfirmation}/>

[**forceRefresh: bool**](https://reacttraining.com/web/api/BrowserRouter/forcerefresh-bool)

If true the router will use full page refreshes on page navigation. You probably only want this in [browsers that don’t support the HTML5 history API](http://caniuse.com/#feat=history).

const supportsHistory = 'pushState' in window.history <BrowserRouter forceRefresh={!supportsHistory}/>

[**keyLength: number**](https://reacttraining.com/web/api/BrowserRouter/keylength-number)

The length of location.key. Defaults to 6.

<BrowserRouter keyLength={12}/>

[**children: node**](https://reacttraining.com/web/api/BrowserRouter/children-node)

A [single child element](https://facebook.github.io/react/docs/react-api.html#react.children.only) to render.

[**<HashRouter>**](https://reacttraining.com/web/api/HashRouter)

A [<Router>](https://reacttraining.com/core/api/Router) that uses the hash portion of the URL (i.e. window.location.hash) to keep your UI in sync with the URL.

**IMPORTANT NOTE:** Hash history does not support location.key or location.state. In previous versions we attempted to shim the behavior but there were edge-cases we couldn’t solve. Any code or plugin that needs this behavior won’t work. As this technique is only intended to support legacy browsers, we encourage you to configure your server to work with <BrowserHistory> instead.

import { HashRouter } from 'react-router-dom' <HashRouter> <App/> </HashRouter>

[**basename: string**](https://reacttraining.com/web/api/HashRouter/basename-string)

The base URL for all locations. A properly formatted basename should have a leading slash, but no trailing slash.

<HashRouter basename="/calendar"/> <Link to="/today"/> // renders <a href="#/calendar/today">

[**getUserConfirmation: func**](https://reacttraining.com/web/api/HashRouter/getuserconfirmation-func)

A function to use to confirm navigation. Defaults to using [window.confirm](https://developer.mozilla.org/en-US/docs/Web/API/Window/confirm).

// this is the default behavior const getConfirmation = (message, callback) => { const allowTransition = window.confirm(message) callback(allowTransition) } <HashRouter getUserConfirmation={getConfirmation}/>

[**hashType: string**](https://reacttraining.com/web/api/HashRouter/hashtype-string)

The type of encoding to use for window.location.hash. Available values are:

* "slash" - Creates hashes like #/ and #/sunshine/lollipops
* "noslash" - Creates hashes like # and #sunshine/lollipops
* "hashbang" - Creates [“ajax crawlable”](https://developers.google.com/webmasters/ajax-crawling/docs/learn-more) (deprecated by Google) hashes like #!/ and #!/sunshine/lollipops

Defaults to "slash".

[**children: node**](https://reacttraining.com/web/api/HashRouter/children-node)

A [single child element](https://facebook.github.io/react/docs/react-api.html#react.children.only) to render.

[**<Link>**](https://reacttraining.com/web/api/Link)

Provides declarative, accessible navigation around your application.

import { Link } from 'react-router-dom' <Link to="/about">About</Link>

[**to: string**](https://reacttraining.com/web/api/Link/to-string)

A string representation of the location to link to, created by concatenating the location’s pathname, search, and hash properties.

<Link to='/courses?sort=name'/>

[**to: object**](https://reacttraining.com/web/api/Link/to-object)

An object that can have any of the following properties:

* pathname: A string representing the path to link to.
* search: A string represenation of query parameters.
* hash: A hash to put in the URL, e.g. #a-hash.
* state: State to persist to the location.

<Link to={{ pathname: '/courses', search: '?sort=name', hash: '#the-hash', state: { fromDashboard: true } }}/>

[**replace: bool**](https://reacttraining.com/web/api/Link/replace-bool)

When true, clicking the link will replace the current entry in the history stack instead of adding a new one.

<Link to="/courses" replace />

[**innerRef: function**](https://reacttraining.com/web/api/Link/innerref-function)

Allows access to the underlying ref of the component

const refCallback = node => { // `node` refers to the mounted DOM element or null when unmounted } <Link to="/" innerRef={refCallback} />

[**others**](https://reacttraining.com/web/api/Link/others)

You can also pass props you’d like to be on the <a> such as a title, id, className, etc.

[**<NavLink>**](https://reacttraining.com/web/api/NavLink)

A special version of the [<Link>](https://reacttraining.com/react-router/Link.md) that will add styling attributes to the rendered element when it matches the current URL.

import { NavLink } from 'react-router-dom' <NavLink to="/about">About</NavLink>

[**activeClassName: string**](https://reacttraining.com/web/api/NavLink/activeclassname-string)

The class to give the element when it is active. The default given class is active. This will be joined with the className prop.

<NavLink to="/faq" activeClassName="selected" >FAQs</NavLink>

[**activeStyle: object**](https://reacttraining.com/web/api/NavLink/activestyle-object)

The styles to apply to the element when it is active.

<NavLink to="/faq" activeStyle={{ fontWeight: 'bold', color: 'red' }} >FAQs</NavLink>

[**exact: bool**](https://reacttraining.com/web/api/NavLink/exact-bool)

When true, the active class/style will only be applied if the location is matched exactly.

<NavLink exact to="/profile" >Profile</NavLink>

[**strict: bool**](https://reacttraining.com/web/api/NavLink/strict-bool)

When true, the trailing slash on a location’s pathname will be taken into consideration when determining if the location matches the current URL. See the [<Route strict>](https://reacttraining.com/core/api/Route/strict-bool) documentation for more information.

<NavLink strict to="/events/" >Events</NavLink>

[**isActive: func**](https://reacttraining.com/web/api/NavLink/isactive-func)

A function to add extra logic for determining whether the link is active. This should be used if you want to do more than verify that the link’s pathname matches the current URL’s pathname.

// only consider an event active if its event id is an odd number const oddEvent = (match, location) => { if (!match) { return false } const eventID = parseInt(match.params.eventID) return !isNaN(eventID) && eventID % 2 === 1 } <NavLink to="/events/123" isActive={oddEvent} >Event 123</NavLink>

[**location: object**](https://reacttraining.com/web/api/NavLink/location-object)

The [isActive](https://reacttraining.com/web/api/NavLink/isactive-func) compares the current history location (usually the current browser URL). To compare to a different location, a [location](https://reacttraining.com/core/api/location) can be passed.

[**<Prompt>**](https://reacttraining.com/web/api/Prompt)

Re-exported from core [Prompt](https://reacttraining.com/core/api/Prompt)

[**<MemoryRouter>**](https://reacttraining.com/web/api/MemoryRouter)

A [<Router>](https://reacttraining.com/react-router/Router.md) that keeps the history of your “URL” in memory (does not read or write to the address bar). Useful in tests and non-browser environments like [React Native](https://facebook.github.io/react-native/).

import { MemoryRouter } from 'react-router' <MemoryRouter> <App/> </MemoryRouter>

[**initialEntries: array**](https://reacttraining.com/web/api/MemoryRouter/initialentries-array)

An array of locations in the history stack. These may be full-blown location objects with { pathname, search, hash, state } or simple string URLs.

<MemoryRouter initialEntries={[ '/one', '/two', { pathname: '/three' } ]} initialIndex={1} > <App/> </MemoryRouter>

[**initialIndex: number**](https://reacttraining.com/web/api/MemoryRouter/initialindex-number)

The initial location’s index in the array of initialEntries.

[**getUserConfirmation: func**](https://reacttraining.com/web/api/MemoryRouter/getuserconfirmation-func)

A function to use to confirm navigation. You must use this option when using <MemoryRouter> directly with a <Prompt>.

[**keyLength: number**](https://reacttraining.com/web/api/MemoryRouter/keylength-number)

The length of location.key. Defaults to 6.

<MemoryRouter keyLength={12}/>

[**children: node**](https://reacttraining.com/web/api/MemoryRouter/children-node)

A [single child element](https://facebook.github.io/react/docs/react-api.html#react.children.only) to render.

[**<Redirect>**](https://reacttraining.com/web/api/Redirect)

Rendering a <Redirect> will navigate to a new location. The new location will override the current location in the history stack, like server-side redirects (HTTP 3xx) do.

import { Route, Redirect } from 'react-router' <Route exact path="/" render={() => ( loggedIn ? ( <Redirect to="/dashboard"/> ) : ( <PublicHomePage/> ) )}/>

[**to: string**](https://reacttraining.com/web/api/Redirect/to-string)

The URL to redirect to. Any valid URL path that [path-to-regexp](https://www.npmjs.com/package/path-to-regexp)understands. All URL parameters that are used in to must be covered by from.

<Redirect to="/somewhere/else"/>

[**to: object**](https://reacttraining.com/web/api/Redirect/to-object)

A location to redirect to. pathname can be any valid URL path that [path-to-regexp](https://www.npmjs.com/package/path-to-regexp) understands.

<Redirect to={{ pathname: '/login', search: '?utm=your+face', state: { referrer: currentLocation } }}/>

The state object can be accessed via this.props.location.state in the redirected-to component. This new referrer key (which is not a special name) would then be accessed via this.props.location.state.referrerin the Login component pointed to by the pathname '/login'

[**push: bool**](https://reacttraining.com/web/api/Redirect/push-bool)

When true, redirecting will push a new entry onto the history instead of replacing the current one.

<Redirect push to="/somewhere/else"/>

[**from: string**](https://reacttraining.com/web/api/Redirect/from-string)

A pathname to redirect from. Any valid URL path that [path-to-regexp](https://www.npmjs.com/package/path-to-regexp)understands. All matched URL parameters are provided to the pattern in to. Must contain all parameters that are used in to. Additional parameters not used by to are ignored.

This can only be used to match a location when rendering a <Redirect>inside of a <Switch>. See [<Switch children>](https://reacttraining.com/web/api/Switch/children-node) for more details.

<Switch> <Redirect from='/old-path' to='/new-path'/> <Route path='/new-path' component={Place}/> </Switch>

// Redirect with matched parameters <Switch> <Redirect from='/users/:id' to='/users/profile/:id'/> <Route path='/users/profile/:id' component={Profile}/> </Switch>

[**exact: bool**](https://reacttraining.com/web/api/Redirect/exact-bool)

Match from exactly; equivalent to [Route.exact](https://reacttraining.com/web/api/Route/exact-bool).

[**strict: bool**](https://reacttraining.com/web/api/Redirect/strict-bool)

Match from strictly; equivalent to [Route.strict](https://reacttraining.com/web/api/Route/strict-bool).

[**<Route>**](https://reacttraining.com/web/api/Route)

The Route component is perhaps the most important component in React Router to understand and learn to use well. Its most basic responsibility is to render some UI when a [location](https://reacttraining.com/web/api/location) matches the route’s path.

Consider the following code:

import { BrowserRouter as Router, Route } from 'react-router-dom' <Router> <div> <Route exact path="/" component={Home}/> <Route path="/news" component={NewsFeed}/> </div> </Router>

If the location of the app is / then the UI hierarchy will be something like:

<div> <Home/> <!-- react-empty: 2 --> </div>

And if the location of the app is /news then the UI hierarchy will be:

<div> <!-- react-empty: 1 --> <NewsFeed/> </div>

The “react-empty” comments are just implementation details of React’s nullrendering. But for our purposes, it is instructive. A Route is always technically “rendered” even though its rendering null. As soon as the app location matches the route’s path, your component will be rendered.

[**Route render methods**](https://reacttraining.com/web/api/Route/route-render-methods)

There are 3 ways to render something with a <Route>:

* [<Route component>](https://reacttraining.com/web/api/Route/component)
* [<Route render>](https://reacttraining.com/web/api/Route/render-func)
* [<Route children>](https://reacttraining.com/web/api/Route/children-func)

Each is useful in different circumstances. You should use only one of these props on a given <Route>. See their explanations below to understand why you have 3 options. Most of the time you’ll use component.

[**Route props**](https://reacttraining.com/web/api/Route/route-props)

All three [render methods](https://reacttraining.com/web/api/Route/route-render-methods) will be passed the same three route props

* [match](https://reacttraining.com/web/api/match)
* [location](https://reacttraining.com/web/api/location)
* [history](https://reacttraining.com/web/api/history)

[**component**](https://reacttraining.com/web/api/Route/component)

A React component to render only when the location matches. It will be rendered with [route props](https://reacttraining.com/web/api/Route/route-props).

<Route path="/user/:username" component={User}/> const User = ({ match }) => { return <h1>Hello {match.params.username}!</h1> }

When you use component (instead of render or children, below) the router uses [React.createElement](https://facebook.github.io/react/docs/react-api.html#createelement) to create a new [React element](https://facebook.github.io/react/docs/rendering-elements.html) from the given component. That means if you provide an inline function to the componentprop, you would create a new component every render. This results in the existing component unmounting and the new component mounting instead of just updating the existing component. When using an inline function for inline rendering, use the render or the children prop (below).

[**render: func**](https://reacttraining.com/web/api/Route/render-func)

This allows for convenient inline rendering and wrapping without the undesired remounting explained above.

Instead of having a new [React element](https://facebook.github.io/react/docs/rendering-elements.html) created for you using the [component](https://reacttraining.com/web/api/Route/component)prop, you can pass in a function to be called when the location matches. The render prop receives all the same [route props](https://reacttraining.com/web/api/Route/route-props) as the component render prop.

// convenient inline rendering <Route path="/home" render={() => <div>Home</div>}/> // wrapping/composing const FadingRoute = ({ component: Component, ...rest }) => ( <Route {...rest} render={props => ( <FadeIn> <Component {...props}/> </FadeIn> )}/> ) <FadingRoute path="/cool" component={Something}/>

**Warning:** <Route component> takes precedence over <Route render> so don’t use both in the same <Route>.

[**children: func**](https://reacttraining.com/web/api/Route/children-func)

Sometimes you need to render whether the path matches the location or not. In these cases, you can use the function children prop. It works exactly like render except that it gets called whether there is a match or not.

The children render prop receives all the same [route props](https://reacttraining.com/web/api/Route/route-props) as the component and render methods, except when a route fails to match the URL, then match is null. This allows you to dynamically adjust your UI based on whether or not the route matches. Here we’re adding an active class if the route matches

<ul> <ListItemLink to="/somewhere"/> <ListItemLink to="/somewhere-else"/> </ul> const ListItemLink = ({ to, ...rest }) => ( <Route path={to} children={({ match }) => ( <li className={match ? 'active' : ''}> <Link to={to} {...rest}/> </li> )}/> )

This could also be useful for animations:

<Route children={({ match, ...rest }) => ( {/\* Animate will always render, so you can use lifecycles to animate its child in and out \*/} <Animate> {match && <Something {...rest}/>} </Animate> )}/>

**Warning:** Both <Route component> and <Route render> take precedence over <Route children> so don’t use more than one in the same <Route>.

[**path: string**](https://reacttraining.com/web/api/Route/path-string)

Any valid URL path that [path-to-regexp](https://www.npmjs.com/package/path-to-regexp) understands.

<Route path="/users/:id" component={User}/>

Routes without a path *always* match.

[**exact: bool**](https://reacttraining.com/web/api/Route/exact-bool)

When true, will only match if the path matches the location.pathname*exactly*.

<Route exact path="/one" component={About}/>

| *path* | *location.pathname* | *exact* | *matches?* |
| --- | --- | --- | --- |
| /one | /one/two | true | no |
| /one | /one/two | false | yes |

[**strict: bool**](https://reacttraining.com/web/api/Route/strict-bool)

When true, a path that has a trailing slash will only match a location.pathname with a trailing slash. This has no effect when there are additional URL segments in the location.pathname.

<Route strict path="/one/" component={About}/>

| *path* | *location.pathname* | *matches?* |
| --- | --- | --- |
| /one/ | /one | no |
| /one/ | /one/ | yes |
| /one/ | /one/two | yes |

**Warning:** strict can be used to enforce that a location.pathname has no trailing slash, but in order to do this both strict and exact must be true.

<Route exact strict path="/one" component={About}/>

| *path* | *location.pathname* | *matches?* |
| --- | --- | --- |
| /one | /one | yes |
| /one | /one/ | no |
| /one | /one/two | no |

[**location: object**](https://reacttraining.com/web/api/Route/location-object)

A <Route> element tries to match its path to the current history location (usually the current browser URL). However, a [location](https://reacttraining.com/react-router/location.md) with a different pathname can also be passed for matching.

This is useful in cases when you need to match a <Route> to a location other than the current history location, as shown in the [Animated Transitions](https://reacttraining.com/react-router/web/example/animated-transitions)example.

If a <Route> element is wrapped in a <Switch> and matches the location passed to the <Switch> (or the current history location), then the locationprop passed to <Route> will be overridden by the one used by the <Switch>(given [here](https://github.com/ReactTraining/react-router/blob/master/packages/react-router/modules/Switch.js#L51)).

[**sensitive: bool**](https://reacttraining.com/web/api/Route/sensitive-bool)

When true, will match if the path is **case sensitive**.

<Route sensitive path="/one" component={About}/>

| *path* | *location.pathname* | *sensitive* | *matches?* |
| --- | --- | --- | --- |
| /one | /one | true | yes |
| /One | /one | true | no |
| /One | /one | false | yes |

[**<Router>**](https://reacttraining.com/web/api/Router)

The common low-level interface for all router components. Typically apps will use one of the high-level routers instead:

* [<BrowserRouter>](https://reacttraining.com/web/api/BrowserRouter)
* [<HashRouter>](https://reacttraining.com/web/api/HashRouter)
* [<MemoryRouter>](https://reacttraining.com/web/api/MemoryRouter)
* [<NativeRouter>](https://reacttraining.com/native/api/NativeRouter)
* [<StaticRouter>](https://reacttraining.com/web/api/StaticRouter)

The most common use-case for using the low-level <Router> is to synchronize a custom history with a state management lib like Redux or Mobx. Note that this is not required to use state management libs alongside React Router, it’s only for deep integration.

import { Router } from 'react-router' import createBrowserHistory from 'history/createBrowserHistory' const history = createBrowserHistory() <Router history={history}> <App/> </Router>

[**history: object**](https://reacttraining.com/web/api/Router/history-object)

A [history](https://github.com/ReactTraining/history) object to use for navigation.

import createBrowserHistory from 'history/createBrowserHistory' const customHistory = createBrowserHistory() <Router history={customHistory}/>

[**children: node**](https://reacttraining.com/web/api/Router/children-node)

A [single child element](https://facebook.github.io/react/docs/react-api.html#react.children.only) to render.

<Router> <App/> </Router>

[**<StaticRouter>**](https://reacttraining.com/web/api/StaticRouter)

A [<Router>](https://reacttraining.com/react-router/Router.md) that never changes location.

This can be useful in server-side rendering scenarios when the user isn’t actually clicking around, so the location never actually changes. Hence, the name: static. It’s also useful in simple tests when you just need to plug in a location and make assertions on the render output.

Here’s an example node server that sends a 302 status code for [<Redirect>](https://reacttraining.com/react-router/Redirect.md)s and regular HTML for other requests:

import { createServer } from 'http' import React from 'react' import ReactDOMServer from 'react-dom/server' import { StaticRouter } from 'react-router' createServer((req, res) => { // This context object contains the results of the render const context = {} const html = ReactDOMServer.renderToString( <StaticRouter location={req.url} context={context}> <App/> </StaticRouter> ) // context.url will contain the URL to redirect to if a <Redirect> was used if (context.url) { res.writeHead(302, { Location: context.url }) res.end() } else { res.write(html) res.end() } }).listen(3000)

[**basename: string**](https://reacttraining.com/web/api/StaticRouter/basename-string)

The base URL for all locations. A properly formatted basename should have a leading slash, but no trailing slash.

<StaticRouter basename="/calendar"> <Link to="/today"/> // renders <a href="/calendar/today"> </StaticRouter>

[**location: string**](https://reacttraining.com/web/api/StaticRouter/location-string)

The URL the server received, probably req.url on a node server.

<StaticRouter location={req.url}> <App/> </StaticRouter>

[**location: object**](https://reacttraining.com/web/api/StaticRouter/location-object)

A location object shaped like { pathname, search, hash, state }

<StaticRouter location={{ pathname: '/bubblegum' }}> <App/> </StaticRouter>

[**context: object**](https://reacttraining.com/web/api/StaticRouter/context-object)

A plain JavaScript object. During the render, components can add properties to the object to store information about the render.

const context = {} <StaticRouter context={context}> <App /> </StaticRouter>

When a <Route> matches, it will pass the context object to the component it renders as the staticContext prop. Check out the [Server Rendering guide](https://reacttraining.com/web/guides/server-rendering)for more information on how to do this yourself.

After the render, these properties can be used to to configure the server’s response.

if(context.status === '404') { // ... }

[**children: node**](https://reacttraining.com/web/api/StaticRouter/children-node)

A [single child element](https://facebook.github.io/react/docs/react-api.html#react.children.only) to render.

[**<Switch>**](https://reacttraining.com/web/api/Switch)

Renders the first child [<Route>](https://reacttraining.com/react-router/Route.md) or [<Redirect>](https://reacttraining.com/react-router/Redirect.md) that matches the location.

**How is this different than just using a bunch of <Route>s?**

<Switch> is unique in that it renders a route *exclusively*. In contrast, every <Route> that matches the location renders *inclusively*. Consider this code:

<Route path="/about" component={About}/> <Route path="/:user" component={User}/> <Route component={NoMatch}/>

If the URL is /about, then <About>, <User>, and <NoMatch> will all render because they all match the path. This is by design, allowing us to compose <Route>s into our apps in many ways, like sidebars and breadcrumbs, bootstrap tabs, etc.

Occasionally, however, we want to pick only one <Route> to render. If we’re at /about we don’t want to also match /:user (or show our “404” page). Here’s how to do it with Switch:

import { Switch, Route } from 'react-router' <Switch> <Route exact path="/" component={Home}/> <Route path="/about" component={About}/> <Route path="/:user" component={User}/> <Route component={NoMatch}/> </Switch>

Now, if we’re at /about, <Switch> will start looking for a matching <Route>. <Route path="/about"/> will match and <Switch> will stop looking for matches and render <About>. Similarly, if we’re at /michael then <User> will render.

This is also useful for animated transitions since the matched <Route> is rendered in the same position as the previous one.

<Fade> <Switch> {/\* there will only ever be one child here \*/} <Route/> <Route/> </Switch> </Fade> <Fade> <Route/> <Route/> {/\* there will always be two children here, one might render null though, making transitions a bit more cumbersome to work out \*/} </Fade>

[**location: object**](https://reacttraining.com/web/api/Switch/location-object)

A [location](https://reacttraining.com/web/api/location) object to be used for matching children elements instead of the current history location (usually the current browser URL).

[**children: node**](https://reacttraining.com/web/api/Switch/children-node)

All children of a <Switch> should be <Route> or <Redirect> elements. Only the first child to match the current location will be rendered.

<Route> elements are matched using their path prop and <Redirect>elements are matched using their from prop. A <Route> with no path prop or a <Redirect> with no from prop will always match the current location.

When you include a <Redirect> in a <Switch>, it can use any of the <Route>'s location matching props: path, exact, and strict. from is just an alias for the path prop.

If a location prop is given to the <Switch>, it will override the locationprop on the matching child element.

<Switch> <Route exact path="/" component={Home}/> <Route path="/users" component={Users}/> <Redirect from="/accounts" to="/users"/> <Route component={NoMatch}/> </Switch>

[**history**](https://reacttraining.com/web/api/history)

The term “history” and "history object" in this documentation refers to [the history package](https://github.com/ReactTraining/history), which is one of only 2 major dependencies of React Router (besides React itself), and which provides several different implementations for managing session history in JavaScript in various environments.

The following terms are also used:

* “browser history” - A DOM-specific implementation, useful in web browsers that support the HTML5 history API
* “hash history” - A DOM-specific implementation for legacy web browsers
* “memory history” - An in-memory history implementation, useful in testing and non-DOM environments like React Native

history objects typically have the following properties and methods:

* length - (number) The number of entries in the history stack
* action - (string) The current action (PUSH, REPLACE, or POP)
* location - (object) The current location. May have the following properties:
  + pathname - (string) The path of the URL
  + search - (string) The URL query string
  + hash - (string) The URL hash fragment
  + state - (object) location-specific state that was provided to e.g. push(path, state) when this location was pushed onto the stack. Only available in browser and memory history.
* push(path, [state]) - (function) Pushes a new entry onto the history stack
* replace(path, [state]) - (function) Replaces the current entry on the history stack
* go(n) - (function) Moves the pointer in the history stack by n entries
* goBack() - (function) Equivalent to go(-1)
* goForward() - (function) Equivalent to go(1)
* block(prompt) - (function) Prevents navigation (see [the history docs](https://github.com/ReactTraining/history#blocking-transitions))

[**history is mutable**](https://reacttraining.com/web/api/history/history-is-mutable)

The history object is mutable. Therefore it is recommended to access the [location](https://reacttraining.com/web/api/location) from the render props of [<Route>](https://reacttraining.com/web/api/Route), not from history.location. This ensures your assumptions about React are correct in lifecycle hooks. For example:

class Comp extends React.Component { componentWillReceiveProps(nextProps) { // will be true const locationChanged = nextProps.location !== this.props.location // INCORRECT, will \*always\* be false because history is mutable. const locationChanged = nextProps.history.location !== this.props.history.location } } <Route component={Comp}/>

Additional properties may also be present depending on the implementation you’re using. Please refer to [the history documentation](https://github.com/ReactTraining/history#properties) for more details.

[**location**](https://reacttraining.com/web/api/location)

Locations represent where the app is now, where you want it to go, or even where it was. It looks like this:

{ key: 'ac3df4', // not with HashHistory! pathname: '/somewhere' search: '?some=search-string', hash: '#howdy', state: { [userDefined]: true } }

The router will provide you with a location object in a few places:

* [Route component](https://reacttraining.com/web/api/Route/component) as this.props.location
* [Route render](https://reacttraining.com/web/api/Route/render-func) as ({ location }) => ()
* [Route children](https://reacttraining.com/web/api/Route/children-func) as ({ location }) => ()
* [withRouter](https://reacttraining.com/web/api/withRouter) as this.props.location

It is also found on history.location but you shouldn’t use that because its mutable. You can read more about that in the [history](https://reacttraining.com/web/api/history) doc.

A location object is never mutated so you can use it in the lifecycle hooks to determine when navigation happens, this is really useful for data fetching and animation.

componentWillReceiveProps(nextProps) { if (nextProps.location !== this.props.location) { // navigated! } }

You can provide locations instead of strings to the various places that navigate:

* Web [Link to](https://reacttraining.com/web/api/Link/to)
* Native [Link to](https://reacttraining.com/native/api/Link/to)
* [Redirect to](https://reacttraining.com/web/api/Redirect/to)
* [history.push](https://reacttraining.com/web/api/history/push)
* [history.replace](https://reacttraining.com/web/api/history/push)

Normally you just use a string, but if you need to add some “location state” that will be available whenever the app returns to that specific location, you can use a location object instead. This is useful if you want to branch UI based on navigation history instead of just paths (like modals).

// usually all you need <Link to="/somewhere"/> // but you can use a location instead const location = { pathname: '/somewhere', state: { fromDashboard: true } } <Link to={location}/> <Redirect to={location}/> history.push(location) history.replace(location)

Finally, you can pass a location to the following components:

* [Route](https://reacttraining.com/web/api/Route/location)
* [Switch](https://reacttraining.com/web/api/Switch/location)

This will prevent them from using the actual location in the router’s state. This is useful for animation and pending navigation, or any time you want to trick a component into rendering at a different location than the real one.

[**match**](https://reacttraining.com/web/api/match)

A match object contains information about how a <Route path> matched the URL. match objects contain the following properties:

* params - (object) Key/value pairs parsed from the URL corresponding to the dynamic segments of the path
* isExact - (boolean) true if the entire URL was matched (no trailing characters)
* path - (string) The path pattern used to match. Useful for building nested <Route>s
* url - (string) The matched portion of the URL. Useful for building nested <Link>s

You’ll have access match objects in various places:

* [Route component](https://reacttraining.com/web/api/Route/component) as this.props.match
* [Route render](https://reacttraining.com/web/api/Route/render-func) as ({ match }) => ()
* [Route children](https://reacttraining.com/web/api/Route/children-func) as ({ match }) => ()
* [withRouter](https://reacttraining.com/web/api/withRouter) as this.props.match
* [matchPath](https://reacttraining.com/web/api/matchPath) as the return value

If a Route does not have a path, and therefore always matches, you’ll get the closest parent match. Same goes for withRouter.

[**null matches**](https://reacttraining.com/web/api/match/null-matches)

A <Route> that uses the children prop will call its children function even when the route’s path does not match the current location. When this is the case, the match will be null. Being able to render a <Route>'s contents when it does match can be useful, but certain challenges arise from this situation.

The default way to “resolve” URLs is to join the match.url string to the “relative” path.

`${match.url}/relative-path`

If you attempt to do this when the match is null, you will end up with a TypeError. This means that it is considered unsafe to attempt to join “relative” paths inside of a <Route> when using the children prop.

A similar, but more subtle situation occurs when you use a pathless <Route>inside of a <Route> that generates a null match object.

// location.pathname = '/matches' <Route path='/does-not-match' children={({ match }) => ( // match === null <Route render={({ match:pathlessMatch }) => ( // pathlessMatch === ??? )}/> )}/>

Pathless <Route>s inherit their match object from their parent. If their parent match is null, then their match will also be null. This means that a) any child routes/links will have to be absolute because there is no parent to resolve with and b) a pathless route whose parent match can be null will need to use the children prop to render.

[**matchPath**](https://reacttraining.com/web/api/matchPath)

This lets you use the same matching code that <Route> uses except outside of the normal render cycle, like gathering up data dependencies before rendering on the server.

import { matchPath } from 'react-router' const match = matchPath('/users/123', { path: '/users/:id', exact: true, strict: false })

[**pathname**](https://reacttraining.com/web/api/matchPath/pathname)

The first argument is the pathname you want to match. If you’re using this on the server with Node.js, it would be req.path.

[**props**](https://reacttraining.com/web/api/matchPath/props)

The second argument are the props to match against, they are identical to the matching props Route accepts:

{ path, // like /users/:id strict, // optional, defaults to false exact // optional, defaults to false }

[**withRouter**](https://reacttraining.com/web/api/withRouter)

You can get access to the [history](https://reacttraining.com/web/api/history) object’s properties and the closest [<Route>](https://reacttraining.com/web/api/Route)'s [match](https://reacttraining.com/web/api/match) via the withRouter higher-order component. withRouterwill pass updated match, location, and history props to the wrapped component whenever it renders.

import React from 'react' import PropTypes from 'prop-types' import { withRouter } from 'react-router' // A simple component that shows the pathname of the current location class ShowTheLocation extends React.Component { static propTypes = { match: PropTypes.object.isRequired, location: PropTypes.object.isRequired, history: PropTypes.object.isRequired } render() { const { match, location, history } = this.props return ( <div>You are now at {location.pathname}</div> ) } } // Create a new component that is "connected" (to borrow redux // terminology) to the router. const ShowTheLocationWithRouter = withRouter(ShowTheLocation)

**Important Note**

withRouter does not subscribe to location changes like React Redux’s connect does for state changes. Instead, re-renders after location changes propagate out from the <Router> component. This means that withRouterdoes *not* re-render on route transitions unless its parent component re-renders. If you are using withRouter to prevent updates from being blocked by shouldComponentUpdate, it is important that withRouter wraps the component that implements shouldComponentUpdate. For example, when using Redux:

// This gets around shouldComponentUpdate withRouter(connect(...)(MyComponent)) // or compose( withRouter, connect(...) )(MyComponent) // This does not connect(...)(withRouter(MyComponent)) // nor compose( connect(...), withRouter )(MyComponent)

See [this guide](https://github.com/ReactTraining/react-router/blob/master/packages/react-router/docs/guides/blocked-updates.md) for more information.

**Static Methods and Properties**

All non-react specific static methods and properties of the wrapped component are automatically copied to the "connected" component.

[**Component.WrappedComponent**](https://reacttraining.com/web/api/withRouter/componentwrappedcomponent)

The wrapped component is exposed as the static property WrappedComponent on the returned component, which can be used for testing the component in isolation, among other things.

// MyComponent.js export default withRouter(MyComponent) // MyComponent.test.js import MyComponent from './MyComponent' render(<MyComponent.WrappedComponent location={{...}} ... />)

[**wrappedComponentRef: func**](https://reacttraining.com/web/api/withRouter/wrappedcomponentref-func)

A function that will be passed as the ref prop to the wrapped component.

class Container extends React.Component { componentDidMount() { this.component.doSomething() } render() { return ( <MyComponent wrappedComponentRef={c => this.component = c}/> ) } }