# Package 'reactRouter'

May 10, 2026

**Type** Package

**Title** 'React Router' for 'shiny' Apps and 'Quarto'

**Version** 0.2.0

**Maintainer** Felix Luginbuhl <felix.luginbuhl@protonmail.ch>

**Description** Provides a wrapper around the 'react-router-dom' 'React' library for use in 'Shiny' applications and 'Quarto' documents. Enables client-side routing with hash, memory, and browser history strategies, nested routes, dynamic segments, data loaders, actions, and navigation hooks.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Depends** R (>= 4.0)

**Imports** htmltools, shiny, shiny.react, checkmate, jsonlite

**Suggests** testthat (>= 3.0.0), chromote (>= 0.1.1), shinytest2, dplyr, muiMaterial

**RoxygenNote** 7.3.3

**URL** https://felixluginbuhl.com/reactRouter/

**BugReports** https://github.com/lgnbhl/reactRouter/issues

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Felix Luginbuhl [aut, cre, cph] (ORCID: <https://orcid.org/0009-0008-6625-2899>), Andryas Waurzenczak [ctb]

**Repository** CRAN

**Date/Publication** 2026-05-10 10:20:02 UTC

## Contents

---

`Await` *Await*

---

## Description

https://api.reactrouter.com/v7/functions/react-router.Await.html

## Usage

```
Await(
  into = NULL,
  as = "children",
  resolveKey,
  selector = NULL,
  render = NULL,
  errorElement = NULL,
  fallback = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `into` | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| `as` | Character. The name of the component's prop to inject the hook data into (by default `"children"` for text display, `"rows"` for a data grid, `"value"` for an input). |
| `resolveKey` | Character. The key in the loader's return value that holds the promise (e.g. if the loader returns `{ data: promise }`, set `resolveKey = "data"`). |
| `selector` | Character. Optional key to extract from the hook data object. If `NULL` (the default), the entire data is passed. Dotted paths like `"summary.title"` navigate nested objects. |
| `render` | Optional `JS` function `(value) => ReactNode` used in place of `into`/`as`. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. `JS("v => `${v.first} ${v.last}`")`). |
| `errorElement` | Element to render if the promise rejects. |
| `fallback` | Element shown while the promise is pending. Defaults to a plain `"Loading\u2026"` span. |
| `...` | Additional props to pass to the component. |

### Details

Renders `into` when a deferred loader promise resolves, injecting the resolved value (or a `selector` from it) `as` a prop. Use inside a `Route` whose `loader` returns an object containing a promise (written via `JS`). In React Router v7, simply return the object directly – no `defer()` wrapper is needed.

---

## `BrowserRouter` *BrowserRouter*

---

### Description

https://api.reactrouter.com/v7/functions/react-router.BrowserRouter.html

### Usage

```
BrowserRouter(...)
```

### Arguments

| | |
|---|---|
| ... | Props to pass to element. |

### Value

A BrowserRouter component.

---

## `createBrowserRouter` *createBrowserRouter*

---

### Description

https://api.reactrouter.com/v7/functions/react-router.createBrowserRouter.html

### Usage

```
createBrowserRouter(...)
```

### Arguments

| | |
|---|---|
| ... | `Route` elements. Pass directly, or (optionally) wrapped in `createRoutesFromElements` to mirror the official React Router v7 API. |

### Details

Creates a browser router using the data router API. Use with `createRoutesFromElements` and `Route`.

### Value

A createBrowserRouter component.

createHashRouter *createHashRouter*

## Description

https://api.reactrouter.com/v7/functions/react-router.createHashRouter.html

## Usage

```
createHashRouter(...)
```

## Arguments

| | |
|---|---|
| ... | Route elements. Pass directly, or (optionally) wrapped in createRoutesFromElements to mirror the official React Router v7 API. |

## Details

Creates a hash router using the data router API. Use with createRoutesFromElements and Route.

## Value

A createHashRouter component.

createMemoryRouter *createMemoryRouter*

## Description

https://api.reactrouter.com/v7/functions/react-router.createMemoryRouter.html

## Usage

```
createMemoryRouter(...)
```

## Arguments

| | |
|---|---|
| ... | Route elements. Pass directly, or (optionally) wrapped in createRoutesFromElements to mirror the official React Router v7 API. |

## Details

Creates a memory router using the data router API. Routing state is kept in memory and the browser URL is never read or modified, making it suitable for static HTML pages (file://), Quarto documents, and embedded widgets where the real URL is irrelevant. Use with createRoutesFromElements and Route.

## Value

A createMemoryRouter component.

---

`createRoutesFromElements` *createRoutesFromElements*

---

## Description

https://api.reactrouter.com/v7/variables/react-router.createRoutesFromElements.html

## Usage

```
createRoutesFromElements(...)
```

## Arguments

| | |
|---|---|
| `...` | `Route` elements. |

## Details

Optional compatibility alias. In R, `createHashRouter`, `createBrowserRouter`, and `createMemoryRouter` accept `Route` elements directly, so wrapping them in `createRoutesFromElements()` is not required. The function is kept so that examples copied verbatim from the React Router v7 documentation (`createHashRouter(createRoutesFromElements(...))`) keep working.

The actual JSX-to-route-object conversion always happens on the JavaScript side; this R function simply bundles its arguments into a tag list.

## Value

A tag list of Route elements.

---

`dataResponse` *dataResponse (loader/action helper)*

---

## Description

https://api.reactrouter.com/v7/functions/react-router.data.html

## Usage

```
dataResponse(value, init = NULL)
```

### Arguments

| | |
|---|---|
| value | The payload to expose via useLoaderData() / useActionData(). Either an R object (list, vector, data.frame – serialized to JSON), or a JS expression for a JavaScript value. |
| init | Optional. Either a list with status (integer), statusText (character) and/or headers (named list), or a JS expression evaluating to such an object. |

### Details

Returns a JS loader function that resolves to a React Router data() response – a thin wrapper that lets you attach an HTTP status, statusText, and/or headers alongside the loader/action payload while still exposing value via useLoaderData / useActionData.

Use the R helper for static loaders that always return the same value plus status. For values computed inside a custom loader/action, call window.jsmodule['@/reactRouter'].helpers.data(value, init) directly in your JS() string, e.g.

```
loader = JS("async () => {
  const { data } = window.jsmodule['@/reactRouter'].helpers;
  const rows = await fetchRows();
  return data({ rows }, { status: 200 });
}")
```

### Value

A JS expression suitable for the loader or action argument of Route.

### Examples

```
## Not run:
Route(
  path = "/profile",
  loader = dataResponse(
    list(name = "Ada", role = "Engineer"),
    init = list(status = 200)
  ),
  element = useLoaderData(tags$pre())
)

## End(Not run)
```

---

## Form *Form*

---

### Description

https://api.reactrouter.com/v7/variables/react-router.Form.html

### Usage

```
Form(...)
```

### Arguments

| | |
|---|---|
| ... | Props to pass to element. |

### Value

A Form component.

---

## HashRouter *HashRouter*

---

### Description

https://api.reactrouter.com/v7/functions/react-router.HashRouter.html

### Usage

```
HashRouter(...)
```

### Arguments

| | |
|---|---|
| ... | Props to pass to element. |

### Value

A HashRouter component.

---

## isRouteErrorResponse *isRouteErrorResponse*

---

### Description

https://api.reactrouter.com/v7/functions/react-router.isRouteErrorResponse.html

### Usage

```
isRouteErrorResponse()
```

## Details

Returns a JS reference to the `isRouteErrorResponse` type guard. Use it inside an `errorElement` render callback to branch on whether the error came from a thrown `Response` (e.g. `throw new Response(..., { status: 404 })`) or from arbitrary code. Pair with `useRouteError`.

Calling `isRouteErrorResponse()` from R returns a JS expression that evaluates, in the browser, to the upstream `isRouteErrorResponse` function. Interpolate it inside the `render` string of `useRouteError()` as shown below.

For convenience, the same function is also reachable inside any user-authored JS string as `window.jsmodule['@/reactRoute`

## Value

A JS expression evaluating to the `isRouteErrorResponse` function reference.

## Examples

```
## Not run:
useRouteError(render = JS(paste0(
  "e => ", isRouteErrorResponse(),
  "(e) ? <p>HTTP {e.status}</p> : <p>Unknown error</p>"
)))

## End(Not run)
```

---

`Link` *Link*

---

## Description

https://api.reactrouter.com/v7/variables/react-router.Link.html

https://api.reactrouter.com/v7/variables/react-router.Link.html

**Repeat clicks.** The Shiny input value is the link's `to` string. Clicking the same link twice publishes the same value, and Shiny suppresses identical-value updates by default — your `observeEvent(input$myLink, ...)` will fire only on the first click. If you need to react to every click (e.g. logging, refreshing a panel), bind to a counter or use `shiny::observeEvent(..., ignoreNULL = FALSE, priority = "event")` alongside an explicit click counter, or wrap the click target in a regular `shiny::actionButton()` that triggers the navigation programmatically.

## Usage

```
Link(..., reloadDocument = FALSE)

Link.shinyInput(inputId, ..., reloadDocument = FALSE)

updateLink.shinyInput(
  session = shiny::getDefaultReactiveDomain(),
```

```
  inputId,
  ...
)
```

## Arguments

| | |
|---|---|
| `...` | Props to pass to element. |
| `reloadDocument` | Boolean. Default FALSE. Let browser handle the transition normally |
| `inputId` | ID of the component. |
| `session` | For `updateLink.shinyInput()` / `updateNavLink.shinyInput()` only: the Shiny session object. Defaults to the current reactive domain. Not used by `Link.shinyInput()` / `NavLink.shinyInput()` themselves. |

## Details

The ‘reloadDocument‘ prop controls whether clicking the link uses React Router’s client-side navigation (‘FALSE‘, the default) or skips it and lets the browser handle the click natively (‘TRUE‘). The default is correct for almost every use, including Shiny apps with server-rendered output (‘uiOutput‘, ‘renderUI‘, ‘plotOutput‘, htmlwidgets) — Shiny output bindings re-attach automatically when React Router mounts the new route’s element. See ‘vignette("routers", package = "reactRouter")‘ for details.

**Two flavors.** Pick `Link()` for a plain navigation link (the common case, mirroring React Router’s API one-to-one). Pick `Link.shinyInput()` only when you also need the click to fire a Shiny input on the server — it adds an `inputId` that updates with the link’s `to` every time it is clicked, while still navigating. If in doubt, use `Link()`.

## Value

A Link component.

---

`MemoryRouter` *MemoryRouter*

---

## Description

https://api.reactrouter.com/v7/functions/react-router.MemoryRouter.html

## Usage

```
MemoryRouter(...)
```

## Arguments

| | |
|---|---|
| `...` | Props to pass to element. |

## Value

A MemoryRouter component.

## Navigate *Navigate*

### Description

https://api.reactrouter.com/v7/functions/react-router.Navigate.html

### Usage

```
Navigate(...)
```

### Arguments

| | |
|---|---|
| ... | Props to pass to element. |

### Value

A Navigate component.

## NavLink *NavLink*

### Description

https://api.reactrouter.com/v7/variables/react-router.NavLink.html

https://api.reactrouter.com/v7/variables/react-router.NavLink.html

**Repeat clicks.** The Shiny input value is the link's `to` string; clicking the same link twice publishes the same value and Shiny suppresses identical-value updates by default. See `Link.shinyInput` for the workaround.

### Usage

```
NavLink(..., reloadDocument = FALSE)

NavLink.shinyInput(inputId, ..., reloadDocument = FALSE)

updateNavLink.shinyInput(
  session = shiny::getDefaultReactiveDomain(),
  inputId,
  ...
)
```

### Arguments

| | |
|---|---|
| `...` | Props to pass to element. |
| `reloadDocument` | Boolean. Default FALSE Let browser handle the transition normally |
| `inputId` | ID of the component. |
| `session` | For `updateLink.shinyInput()` / `updateNavLink.shinyInput()` only: the Shiny session object. Defaults to the current reactive domain. Not used by `Link.shinyInput()` / `NavLink.shinyInput()` themselves. |

### Details

The 'reloadDocument' prop controls whether clicking the link uses React Router's client-side navigation ('FALSE', the default) or skips it and lets the browser handle the click natively ('TRUE'). The default is correct for almost every use, including Shiny apps with server-rendered output ('uiOutput', 'renderUI', 'plotOutput', htmlwidgets) — Shiny output bindings re-attach automatically when React Router mounts the new route's element. See 'vignette("routers", package = "reactRouter")' for details.

### Value

A NavLink component.

---

## `Outlet` *Outlet*

---

### Description

https://api.reactrouter.com/v7/functions/react-router.Outlet.html

### Usage

```
Outlet(...)
```

### Arguments

| | |
|---|---|
| `...` | Props to pass to element. |

### Value

A Outlet component.

## print.reactRouter *Print reactRouter components*

### Description

When called interactively, renders the component in the IDE viewer panel. Otherwise, falls back to standard shiny.tag printing (raw HTML text).

### Usage

```
## S3 method for class 'reactRouter'
print(x, browse = interactive(), ...)
```

### Arguments

| | |
|---|---|
| x | A reactRouter object (also inherits shiny.tag). |
| browse | Whether to render in viewer. Defaults to TRUE in interactive sessions. |
| ... | Additional arguments passed to print. |

### Details

Only the router-root constructors carry the "reactRouter" S3 class and therefore dispatch to this method: RouterProvider, createHashRouter, createBrowserRouter, createMemoryRouter, HashRouter, BrowserRouter, and MemoryRouter. Inner pieces (Route, Link, Outlet, hooks, ...) are plain shiny.tag elements – printing one of those on its own is rarely useful (it has no router context to render against), so they intentionally fall through to the default shiny.tag print method.

### Value

Invisibly returns x.

## reactRouterDependency *react-router-dom JS dependency*

### Description

react-router-dom JS dependency

### Usage

```
reactRouterDependency()
```

### Value

HTML dependency object.

## reactRouterExample *Run reactRouterExample example*

### Description

Launch a Shiny example app or list the available examples. Use ‘reactRouter::reactRouterExample("basic")‘ to run a showcase app.

### Usage

```
reactRouterExample(example = NULL, ...)
```

### Arguments

| | |
|---|---|
| example | The name of the example to run, or ‘NULL‘ to print and invisibly return the list of available examples. |
| ... | Additional arguments to pass to ‘shiny::runApp()‘. |

### Value

When ‘example‘ is ‘NULL‘, invisibly returns a character vector of example names (also printed via ‘message()‘). Otherwise this function normally does not return; interrupt R to stop the application (usually by pressing Ctrl+C or Esc).

### See Also

`shiny.blueprint::runExample()`, which this function is adapted from.

## redirect *redirect (loader/action helper)*

### Description

https://reactrouter.com/api/utils/redirect

### Usage

```
redirect(to)
```

### Arguments

| | |
|---|---|
| to | Character. Destination path. |

### Details

Returns a JS loader function that redirects to to. Pass as the loader argument of a Route to perform an unconditional redirect – typically used for guard routes that always send the user somewhere else.

**Security:** to must be a trusted, package-author-controlled string. javascript:, data:, and vbscript: URL schemes are rejected. If you build to from user-supplied input, validate it yourself before passing it in – never round-trip untrusted strings through redirect() into a navigation.

For conditional redirects inside a custom loader/action, call window.jsmodule['@/reactRouter'].helpers.redirect(to from your own JS() string, e.g.

```
loader = JS(
  "async () => {
     const { redirect } = window.jsmodule['@/reactRouter'].helpers;
     if (!authed()) return redirect('/login'); ...
   }"
)
```

The data, replace, and redirectDocument helpers are exposed on the same namespace.

### Value

A JS expression suitable for the loader argument of Route.

### Examples

```
## Not run:
Route(path = "/old", loader = redirect("/new"), element = NULL)

## End(Not run)
```

---

## redirectDocument *redirectDocument (loader/action helper)*

---

### Description

https://reactrouter.com/api/utils/redirectDocument

### Usage

```
redirectDocument(to)
```

### Arguments

| | |
|---|---|
| to | Character. Destination path or absolute URL. |

### Details

Returns a JS loader function that performs a *document* redirect to to – a full page reload, as opposed to the client-side navigation that redirect performs. Use when navigating to a URL outside the router's control (e.g. a server-rendered page) so the browser fully unloads the SPA.

For conditional document redirects inside a custom loader/action, call window.jsmodule['@/reactRouter'].helpers.red from your own JS() string.

### Value

A JS expression suitable for the loader argument of Route.

### Examples

```
## Not run:
Route(
  path = "/docs",
  loader = redirectDocument("/static/docs/index.html"),
  element = NULL
)

## End(Not run)
```

---

## replaceResponse *replaceResponse (loader/action helper)*

---

### Description

https://reactrouter.com/api/utils/replace

### Usage

```
replaceResponse(to)
```

### Arguments

| | |
|---|---|
| to | Character. Destination path. |

### Details

Returns a JS loader function that performs a *replace* navigation to to – same as redirect, but the new entry replaces the current one in the history stack instead of pushing a new one. Use for "alias" routes where the original URL should not remain in the user's back-history.

Renamed from replace() to avoid masking base::replace. This mirrors the dataResponse() naming for the same reason.

For conditional replacements inside a custom loader/action, call window.jsmodule['@/reactRouter'].helpers.replace from your own JS() string.

## Value

A JS expression suitable for the loader argument of Route.

## Examples

```
## Not run:
Route(path = "/legacy", loader = replaceResponse("/new"), element = NULL)

## End(Not run)
```

---

Route *Route*

---

## Description

https://api.reactrouter.com/v7/functions/react-router.Route.html

## Usage

```
Route(
  ...,
  element,
  loader = NULL,
  action = NULL,
  errorElement = NULL,
  key = randomKey()
)
```

## Arguments

| | |
|---|---|
| ... | Additional Route props (see Details). |
| element | The element to render when the route matches. Wrapped on the JS side in a no-DOM Keyed component so React remounts the subtree on every route change – this is required for Shiny output bindings (e.g. textOutput()) to reinitialise correctly when two routes render the same component shape with different namespaces. Unlike the previous  wrapper, Keyed adds no DOM node, so layouts like MUI Grid that require typed direct children keep working. |
| loader | Optional. A JS expression evaluating to a loader function, e.g. JS("({ params }) => fetch(...)"). For a plain unconditional redirect, use redirect. To embed static R data, serialize it first with jsonlite::toJSON() and wrap the result in JS(). |
| action | Optional. A JS expression evaluating to an action function called by Form submissions and useSubmit / useFetcher submits. |
| errorElement | Optional. Element rendered when the route's loader, action, or rendering throws. |

key — Stable React key for the route's element. Defaults to a random alphanumeric string per Route() call, so each route's element has a distinct identity from its siblings – this is what causes React to unmount the previous route's subtree (and reinitialise its Shiny output bindings) when navigating between routes that render the same component shape. You almost never need to pass this explicitly. **Note:** this key only affects identity of the route's element; it does *not* rebuild the RouterProvider data router. The route tree is created once on mount and subsequent Route() edits are ignored; to apply a new route tree at runtime, give RouterProvider itself a changing key (e.g. via shiny::renderUI).

## Details

Internally the 'element' is wrapped in a 'shiny::div()' with a UUID key so, in case R shiny is used, shiny can differentiate each element.

Additional React Router Route props can be passed through ...:

- path (Character): path pattern, supports :param, optional :param?, and splat *.
- index (Boolean): mark this as the index route of its parent.
- caseSensitive (Boolean): match the path case-sensitively.
- id (Character): stable route id, required for use with useRouteLoaderData.
- handle (Any): arbitrary value exposed via useMatches for breadcrumbs and similar use cases.
- shouldRevalidate (JS): function controlling whether the loader re-runs on a given navigation.
- lazy (JS): code-splitting hook returning a Promise resolving to a route module.
- hasErrorBoundary (Boolean): explicit error-boundary flag (rarely needed when errorElement is provided).

## Value

A Route component.

---

# RouterProvider *RouterProvider*

---

## Description

https://api.reactrouter.com/v7/functions/react-router.RouterProvider.html

## Usage

```
RouterProvider(router, fallbackElement = NULL)
```

### Arguments

| | |
|---|---|
| router | A router element produced by createHashRouter, createBrowserRouter, or createMemoryRouter. |
| fallbackElement | Element shown while the initial route's loader is resolving. |

### Details

Renders a data router. Mirrors the React Router v7 composition pattern: pass a router built with createHashRouter, createBrowserRouter, or createMemoryRouter to the router argument.

The underlying router is created once on mount and is not rebuilt when Route() children change at runtime — React Router data routers own their own navigation state and must be stable. To swap the route tree dynamically (e.g. based on Shiny inputs), remount RouterProvider itself, for instance by toggling its parent via shiny::renderUI.

### Value

A RouterProvider component.

---

## Routes *Routes*

---

### Description

https://api.reactrouter.com/v7/functions/react-router.Routes.html

### Usage

```
Routes(...)
```

### Arguments

| | |
|---|---|
| ... | Props to pass to element. |

### Value

A Routes component.

## ScrollRestoration *ScrollRestoration*

### Description

https://api.reactrouter.com/v7/functions/react-router.ScrollRestoration.html

### Usage

```
ScrollRestoration(...)
```

### Arguments

| | |
|---|---|
| ... | Props to pass to element. Notable props: `getKey` (a `JS` function to compute the scroll key from the location) and `storageKey` (Character, custom `sessionStorage` key). |

### Details

Emulates the browser's scroll restoration on location changes after loaders have completed. Place once inside the root layout of a data router app. Requires a data router (`createBrowserRouter`, `createHashRouter`, etc.).

### Value

A ScrollRestoration component.

## useActionData *useActionData*

### Description

https://api.reactrouter.com/v7/functions/react-router.useActionData.html

### Usage

```
useActionData(
  into = NULL,
  as = "children",
  selector = NULL,
  render = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| as | Character. The name of the component's prop to inject the hook data into (by default "children" for text display, "rows" for a data grid, "value" for an input). |
| selector | Character. Optional key to extract from the hook data object. If NULL (the default), the entire data is passed. Dotted paths like "summary.title" navigate nested objects. |
| render | Optional JS function (value) => ReactNode used in place of into/as. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. JS("v => `${v.first} ${v.last}`")). |
| ... | Additional props to pass to the component. |

### Details

Calls the useActionData() hook and injects the result (or a selector from it) as a prop of the into component. Use inside a Route that has an action.

---

## useAsyncError *useAsyncError*

---

### Description

https://api.reactrouter.com/v7/functions/react-router.useAsyncError.html

### Usage

```
useAsyncError(
  into = NULL,
  as = "children",
  selector = NULL,
  render = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| as | Character. The name of the component's prop to inject the hook data into (by default "children" for text display, "rows" for a data grid, "value" for an input). |
| selector | Character. Optional key to extract from the hook data object. If NULL (the default), the entire data is passed. Dotted paths like "summary.title" navigate nested objects. |

| | |
|---|---|
| render | Optional JS function (value) => ReactNode used in place of into/as. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. JS("v => `${v.first} ${v.last}`")). |
| ... | Additional props to pass to the component. |

## Details

Calls the useAsyncError() hook and injects the rejection reason (or a selector from it) as a prop of the into component. Must be rendered inside the errorElement of an Await so the hook can pick up the rejected promise's error.

---

useAsyncValue *useAsyncValue*

---

## Description

https://api.reactrouter.com/v7/functions/react-router.useAsyncValue.html

## Usage

```
useAsyncValue(
  into = NULL,
  as = "children",
  selector = NULL,
  render = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| as | Character. The name of the component's prop to inject the hook data into (by default "children" for text display, "rows" for a data grid, "value" for an input). |
| selector | Character. Optional key to extract from the hook data object. If NULL (the default), the entire data is passed. Dotted paths like "summary.title" navigate nested objects. |
| render | Optional JS function (value) => ReactNode used in place of into/as. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. JS("v => `${v.first} ${v.last}`")). |
| ... | Additional props to pass to the component. |

## Details

Calls the useAsyncValue() hook and injects the resolved value (or a selector from it) as a prop of the into component. Must be rendered inside an Await that has been called in *children mode* (no into / render on Await) – the hook reads the value resolved by the closest <Await> ancestor.

useBlocker *useBlocker*

## Description

https://api.reactrouter.com/v7/functions/react-router.useBlocker.html

## Usage

```
useBlocker(
  into = NULL,
  as = "children",
  selector = "state",
  render = NULL,
  shouldBlock = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| as | Character. The name of the component's prop to inject the hook data into (by default "children" for text display, "rows" for a data grid, "value" for an input). |
| selector | Character. Optional key to extract from the hook data object. If NULL (the default), the entire data is passed. Dotted paths like "summary.title" navigate nested objects. |
| render | Optional JS function (value) => ReactNode used in place of into/as. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. JS("v => `${v.first} ${v.last}`")). |
| shouldBlock | Either FALSE (the default, disables blocking) or a JS function receiving { currentLocation, nextLocation, historyAction } and returning true to block navigation or false to allow it. **Must be a JS() expression**, not an R function – R functions cannot be invoked from inside React Router's blocker callback. |
| ... | Additional props to pass to the component. |

## Details

Calls the useBlocker() hook and injects the blocker's state (or another selector field) as a prop of the into component. Use to intercept navigation – e.g. warn the user about unsaved changes before they leave a route.

The blocker state is one of "unblocked" (default), "blocked" (navigation intercepted), or "proceeding" (user confirmed, navigation in progress).

useFetcher *useFetcher*

## Description

https://api.reactrouter.com/v7/functions/react-router.useFetcher.html

## Usage

```
useFetcher(
  into = NULL,
  as = "children",
  selector = "state",
  render = NULL,
  fetcherKey = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| as | Character. The name of the component's prop to inject the hook data into (by default "children" for text display, "rows" for a data grid, "value" for an input). |
| selector | Character. Optional key to extract from the hook data object. If NULL (the default), the entire data is passed. Dotted paths like "summary.title" navigate nested objects. |
| render | Optional JS function (value) => ReactNode used in place of into/as. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. JS("v => `${v.first} ${v.last}`")). |
| fetcherKey | Character. Optional key to share a fetcher across components (e.g. "my-fetcher"). |
| ... | Additional props to pass to the component. |

## Details

Calls the useFetcher() hook and injects the result (or a selector from it) as a prop of the into component. Use to fetch data or submit forms without causing a navigation. The fetcher object has state ("idle"/"loading"/"submitting") and data (the response from a loader or action).

selector defaults to "state" so the default into/ children display shows a readable string ("idle" / "loading" / "submitting"). The full fetcher object contains methods (submit, load, Form) that would be silently dropped by JSON serialization if the whole object were rendered as children. To call those methods, use the render = JS(...) form, which receives the full fetcher: render = JS("f => <button onClick={() => f.load('/data')}>Reload</button>").

useFetchers *useFetchers*

## Description

https://api.reactrouter.com/v7/functions/react-router.useFetchers.html

## Usage

```
useFetchers(into = NULL, as = "children", selector = NULL, render = NULL, ...)
```

## Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| as | Character. The name of the component's prop to inject the hook data into (by default "children" for text display, "rows" for a data grid, "value" for an input). |
| selector | Character. Optional key to extract from the hook data object. If NULL (the default), the entire data is passed. Dotted paths like "summary.title" navigate nested objects. |
| render | Optional JS function (value) => ReactNode used in place of into/as. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. JS("v => `${v.first} ${v.last}`")). |
| ... | Additional props to pass to the component. |

## Details

Calls the useFetchers() hook and injects the result (or a selector mapped over each fetcher) as a prop of the into component. Returns an array of all active fetchers. Useful for showing a global loading indicator.

useHref *useHref*

## Description

https://api.reactrouter.com/v7/functions/react-router.useHref.html

## Usage

```
useHref(into = NULL, as = "children", to, render = NULL, ...)
```

### Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook value as the specified prop. |
| as | Character. The name of the component's prop to inject the hook value into. Defaults to "children". |
| to | Character. The path to resolve. |
| render | Optional JS function (value) => ReactNode used in place of into/as. |
| ... | Additional props to pass to the component. |

### Details

Calls the useHref() hook and injects the resolved href string as a prop of the into component.

---

useInRouterContext *useInRouterContext*

---

### Description

https://api.reactrouter.com/v7/functions/react-router.useInRouterContext.html

### Usage

```
useInRouterContext(into = NULL, as = "children", render = NULL, ...)
```

### Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook value as the specified prop. |
| as | Character. The name of the component's prop to inject the hook value into. Defaults to "children". |
| render | Optional JS function (value) => ReactNode used in place of into/as. |
| ... | Additional props to pass to the component. |

### Details

Calls the useInRouterContext() hook and injects the boolean result as a prop of the into component. Useful inside reusable components that may be rendered with or without a surrounding router – guard router-only logic with this check before calling other hooks.

useLinkClickHandler *useLinkClickHandler*

## Description

https://api.reactrouter.com/v7/functions/react-router.useLinkClickHandler.html

## Usage

```
useLinkClickHandler(
  into = NULL,
  as = "children",
  render = NULL,
  to,
  replace = NULL,
  state = NULL,
  target = NULL,
  preventScrollReset = NULL,
  relative = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook value as the specified prop. |
| as | Character. The name of the component's prop to inject the hook value into. Defaults to "children". |
| render | Optional JS function (value) => ReactNode used in place of into/as. |
| to | Character. Destination path. |
| replace | Optional boolean. Replace the current entry in the history stack instead of pushing a new one. |
| state | Optional. State value to attach to the new location. |
| target | Optional character. Anchor target (e.g. "_blank"). |
| preventScrollReset | Optional boolean. If TRUE, do not reset scroll position on navigation. |
| relative | Optional character. Either "route" (default) or "path". |
| ... | Additional props to pass to the component. |

## Details

Calls the useLinkClickHandler() hook and exposes the returned click handler function via render (or injects it as a prop of into, e.g. as = "onClick"). Lets you build link-like components that drive client-side navigation without using Link.

Because the hook returns a function, the render form is the natural fit:

```
  useLinkClickHandler(
    to = "/about",
    render = JS("h => <span onClick={h} role='link'>About</span>")
  )
```

---

useLoaderData *useLoaderData*

---

## Description

https://api.reactrouter.com/v7/functions/react-router.useLoaderData.html

## Usage

```
useLoaderData(
  into = NULL,
  as = "children",
  selector = NULL,
  render = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| as | Character. The name of the component's prop to inject the hook data into (by default "children" for text display, "rows" for a data grid, "value" for an input). |
| selector | Character. Optional key to extract from the hook data object. If NULL (the default), the entire data is passed. Dotted paths like "summary.title" navigate nested objects. |
| render | Optional JS function (value) => ReactNode used in place of into/as. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. JS("v => `${v.first} ${v.last}`")). |
| ... | Additional props to pass to the component. |

## Details

Calls the useLoaderData() hook and injects the result (or a selector from it) as a prop of the into component. Use inside a Route that has a loader.

## Examples

```
## Not run:
# Display a selector as text
useLoaderData(tags$h3(), selector = "name")

# Pass an array to a data grid
useLoaderData(
  muiDataGrid::DataGrid(columns = JS("[
    { field: 'name', headerName: 'Name', flex: 1 }
  ]")),
  as = "rows",
  selector = "people"
)

## End(Not run)
```

---

`useLocation` *useLocation*

---

## Description

https://api.reactrouter.com/v7/functions/react-router.useLocation.html

## Usage

```
useLocation(into = NULL, as = "children", selector = NULL, render = NULL, ...)
```

## Arguments

| | |
|---|---|
| `into` | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| `as` | Character. The name of the component's prop to inject the hook data into (by default `"children"` for text display, `"rows"` for a data grid, `"value"` for an input). |
| `selector` | Character. Optional key to extract from the hook data object. If `NULL` (the default), the entire data is passed. Dotted paths like `"summary.title"` navigate nested objects. |
| `render` | Optional `JS` function `(value) => ReactNode` used in place of `into`/`as`. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. `JS("v => `${v.first} ${v.last}`")`). |
| `...` | Additional props to pass to the component. |

## Details

Calls the `useLocation()` hook and injects the result (or a `selector` from it) `as` a prop of the `into` component. Available selectors: `pathname`, `search`, `hash`, `state`, `key`.

useMatch *useMatch*

## Description

https://api.reactrouter.com/v7/functions/react-router.useMatch.html

## Usage

```
useMatch(
  into = NULL,
  as = "children",
  selector = NULL,
  render = NULL,
  pattern,
  ...
)
```

## Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| as | Character. The name of the component's prop to inject the hook data into (by default "children" for text display, "rows" for a data grid, "value" for an input). |
| selector | Character. Optional key to extract from the hook data object. If NULL (the default), the entire data is passed. Dotted paths like "summary.title" navigate nested objects. |
| render | Optional JS function (value) => ReactNode used in place of into/as. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. JS("v => `${v.first} ${v.last}`")). |
| pattern | Character. The path pattern to match against (e.g. "/products/:id"). |
| ... | Additional props to pass to the component. |

## Details

Calls the useMatch() hook and injects the result (or a selector from it) as a prop of the into component. Returns NULL if no match.

useMatches *useMatches*

## Description

https://api.reactrouter.com/v7/functions/react-router.useMatches.html

## Usage

```
useMatches(into = NULL, as = "children", selector = NULL, render = NULL, ...)
```

## Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| as | Character. The name of the component's prop to inject the hook data into (by default "children" for text display, "rows" for a data grid, "value" for an input). |
| selector | Character. Optional key to extract from the hook data object. If NULL (the default), the entire data is passed. Dotted paths like "summary.title" navigate nested objects. |
| render | Optional JS function (value) => ReactNode used in place of into/as. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. JS("v => `${v.first} ${v.last}`")). |
| ... | Additional props to pass to the component. |

## Details

Calls the useMatches() hook and injects the result (or a selector extracted from each match) as a prop of the into component. Only works inside a data router.

useNavigate *useNavigate*

## Description

https://api.reactrouter.com/v7/functions/react-router.useNavigate.html

## Usage

```
useNavigate(into = NULL, as = "children", render = NULL, ...)
```

### Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook value as the specified prop. |
| as | Character. The name of the component's prop to inject the hook value into. Defaults to "children". |
| render | Optional JS function (value) => ReactNode used in place of into/as. |
| ... | Additional props to pass to the component. |

### Details

Calls the useNavigate() hook and passes the navigate function to render (or injects it as a prop of into). The navigate function has signature navigate(to, options?), e.g. navigate("/about") or navigate(-1) to go back.

Because the hook returns a function (not a value), the render form is the natural way to use it:

```
useNavigate(render = JS(
  "nav => <button onClick={() => nav('/about')}>Go</button>"
))
```

---

## useNavigation *useNavigation*

---

### Description

https://api.reactrouter.com/v7/functions/react-router.useNavigation.html

### Usage

```
useNavigation(
  into = NULL,
  as = "children",
  selector = NULL,
  render = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| as | Character. The name of the component's prop to inject the hook data into (by default "children" for text display, "rows" for a data grid, "value" for an input). |
| selector | Character. Optional key to extract from the hook data object. If NULL (the default), the entire data is passed. Dotted paths like "summary.title" navigate nested objects. |

render — Optional JS function (value) => ReactNode used in place of into/as. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. JS("v => `${v.first} ${v.last}`")).

... — Additional props to pass to the component.

## Details

Calls the useNavigation() hook and injects the result (or a selector from it) as a prop of the into component. Returns the current navigation state: "idle", "loading", or "submitting". Only works inside a data router.

---

useNavigationType *useNavigationType*

---

## Description

https://api.reactrouter.com/v7/functions/react-router.useNavigationType.html

## Usage

```
useNavigationType(into = NULL, as = "children", render = NULL, ...)
```

## Arguments

into — A component (HTML tag or **shiny.react**-based element) that will receive the hook value as the specified prop.

as — Character. The name of the component's prop to inject the hook value into. Defaults to "children".

render — Optional JS function (value) => ReactNode used in place of into/as.

... — Additional props to pass to the component.

## Details

Calls the useNavigationType() hook and injects the result as a prop of the into component. Returns one of "POP", "PUSH", or "REPLACE".

useOutlet *useOutlet*

## Description

https://api.reactrouter.com/v7/functions/react-router.useOutlet.html

## Usage

```
useOutlet(into = NULL, as = "children", render = NULL, context = NULL, ...)
```

## Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook value as the specified prop. |
| as | Character. The name of the component's prop to inject the hook value into. Defaults to "children". |
| render | Optional JS function (value) => ReactNode used in place of into/as. |
| context | Optional value to expose to descendants via useOutletContext. |
| ... | Additional props to pass to the component. |

## Details

Calls the useOutlet() hook and injects the matched child route's element as a prop of the into component (or passes it to render). Returns NULL when no child route matches – useful for rendering a fallback inside a layout when the user is on the parent route.

Differs from the Outlet component in that it returns the element as a value, so you can branch on whether a child route is matched.

## Examples

```
## Not run:
# In a layout route: render the matched child, or a fallback if on the
# parent route itself.
useOutlet(
  render = JS("o => o || <p>Pick a section from the menu.</p>")
)

# Inject the matched outlet element as the body of a wrapping <section>.
useOutlet(into = shiny::tags$section(class = "page"))

## End(Not run)
```

useOutletContext *useOutletContext*

## Description

https://api.reactrouter.com/v7/functions/react-router.useOutletContext.html

## Usage

```
useOutletContext(
  into = NULL,
  as = "children",
  selector = NULL,
  render = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| as | Character. The name of the component's prop to inject the hook data into (by default "children" for text display, "rows" for a data grid, "value" for an input). |
| selector | Character. Optional key to extract from the hook data object. If NULL (the default), the entire data is passed. Dotted paths like "summary.title" navigate nested objects. |
| render | Optional JS function (value) => ReactNode used in place of into/as. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. JS("v => `${v.first} ${v.last}`")). |
| ... | Additional props to pass to the component. |

## Details

Calls the useOutletContext() hook and injects the context value (or a selector from it) as a prop of the into component. The context is whatever was passed to the parent route's Outlet(context = ...) call.

## useParams *useParams*

### Description

https://api.reactrouter.com/v7/functions/react-router.useParams.html

### Usage

```
useParams(into = NULL, as = "children", selector = NULL, render = NULL, ...)
```

### Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| as | Character. The name of the component's prop to inject the hook data into (by default "children" for text display, "rows" for a data grid, "value" for an input). |
| selector | Character. Optional key to extract from the hook data object. If NULL (the default), the entire data is passed. Dotted paths like "summary.title" navigate nested objects. |
| render | Optional JS function (value) => ReactNode used in place of into/as. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. JS("v => `${v.first} ${v.last}`")). |
| ... | Additional props to pass to the component. |

### Details

Calls the useParams() hook and injects the result (or a selector from it) as a prop of the into component. Returns the dynamic parameters from the current URL matched by the Route path pattern.

## useResolvedPath *useResolvedPath*

### Description

https://api.reactrouter.com/v7/functions/react-router.useResolvedPath.html

## Usage

```
useResolvedPath(
  into = NULL,
  as = "children",
  selector = NULL,
  render = NULL,
  to,
  ...
)
```

## Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| as | Character. The name of the component's prop to inject the hook data into (by default "children" for text display, "rows" for a data grid, "value" for an input). |
| selector | Character. Optional key to extract from the hook data object. If NULL (the default), the entire data is passed. Dotted paths like "summary.title" navigate nested objects. |
| render | Optional JS function (value) => ReactNode used in place of into/as. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. JS("v => `${v.first} ${v.last}`")). |
| to | Character. The path to resolve. |
| ... | Additional props to pass to the component. |

## Details

Calls the useResolvedPath() hook and injects the result (or a selector from it) as a prop of the into component. Returns pathname, search, and hash.

---

useRevalidator *useRevalidator*

---

## Description

https://api.reactrouter.com/v7/functions/react-router.useRevalidator.html

## Usage

```
useRevalidator(
  into = NULL,
  as = "children",
  selector = "state",
  render = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| as | Character. The name of the component's prop to inject the hook data into (by default "children" for text display, "rows" for a data grid, "value" for an input). |
| selector | Character. Optional key to extract from the hook data object. If NULL (the default), the entire data is passed. Dotted paths like "summary.title" navigate nested objects. |
| render | Optional JS function (value) => ReactNode used in place of into/as. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. JS("v => `${v.first} ${v.last}`")). |
| ... | Additional props to pass to the component. |

### Details

Calls the useRevalidator() hook and injects the result (or a selector from it) as a prop of the into component. Returns the revalidation state ("idle" or "loading"). Useful for showing loading feedback during manual or polling revalidation.

---

## useRouteError *useRouteError*

---

### Description

https://api.reactrouter.com/v7/functions/react-router.useRouteError.html

### Usage

```
useRouteError(
  into = NULL,
  as = "children",
  selector = NULL,
  render = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| as | Character. The name of the component's prop to inject the hook data into (by default "children" for text display, "rows" for a data grid, "value" for an input). |

| | |
|---|---|
| selector | Character. Optional key to extract from the hook data object. If NULL (the default), the entire data is passed. Dotted paths like "summary.title" navigate nested objects. |
| render | Optional JS function (value) => ReactNode used in place of into/as. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. JS("v => `${v.first} ${v.last}`")). |
| ... | Additional props to pass to the component. |

## Details

Calls the useRouteError() hook and injects the result (or a selector from it) as a prop of the into component. Use inside the errorElement of a Route.

---

useRouteLoaderData *useRouteLoaderData*

---

## Description

https://api.reactrouter.com/v7/functions/react-router.useRouteLoaderData.html

## Usage

```
useRouteLoaderData(
  into = NULL,
  as = "children",
  selector = NULL,
  render = NULL,
  routeId,
  ...
)
```

## Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| as | Character. The name of the component's prop to inject the hook data into (by default "children" for text display, "rows" for a data grid, "value" for an input). |
| selector | Character. Optional key to extract from the hook data object. If NULL (the default), the entire data is passed. Dotted paths like "summary.title" navigate nested objects. |
| render | Optional JS function (value) => ReactNode used in place of into/as. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. JS("v => `${v.first} ${v.last}`")). |
| routeId | Character. The route ID to fetch loader data from. |
| ... | Additional props to pass to the component. |

### Details

Calls the `useRouteLoaderData()` hook and injects the result (or a `selector` from it) `as` a prop of the `into` component. Accesses loader data from any route by its `routeId`. Only works inside a data router.

---

`useRoutes` *useRoutes*

---

### Description

https://api.reactrouter.com/v7/functions/react-router.useRoutes.html

### Usage

```
useRoutes(..., routes = NULL)
```

### Arguments

| | |
|---|---|
| `...` | `Route` elements describing the route tree. Ignored if `routes` is supplied. |
| `routes` | Optional. A `JS` expression evaluating to a plain JavaScript array of route objects (e.g. `JS("[\{ path: '/', element: ... \}]")`), used in place of `Route()` children. |

### Details

Builds a route tree from `Route` children (or a plain object `routes` array) and renders the matched route. The hook-based equivalent of `Routes` / `createRoutesFromElements` for code that prefers a configuration-as-data style. Must be called inside a router (`RouterProvider`, `HashRouter`, etc.).

---

`useSearchParams` *useSearchParams*

---

### Description

https://api.reactrouter.com/v7/functions/react-router.useSearchParams.html

### Usage

```
useSearchParams(into = NULL, as = "children", param = NULL, render = NULL, ...)
```

### Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook data as the specified prop. |
| as | Character. The name of the component's prop to inject the hook data into (by default "children" for text display, "rows" for a data grid, "value" for an input). |
| param | Character. Name of a single query parameter to extract. Returns a character vector of all values for that key (length 0 if absent, length 1+ otherwise). When NULL, returns a named list mapping each key to its vector of values.<br>**Empty vs. missing.** A missing key produces a length-0 vector, which renders as an empty string when injected with the default as = "children". If you need to distinguish "absent" from "present-but-empty" (e.g. show a placeholder), use the render form and branch on Array.isArray(v) && v.length === 0, e.g. render = JS("v => v.length ? v.join(', ') : none"). |
| render | Optional JS function (value) => ReactNode used in place of into/as. Mirrors the native React Router pattern for cases where a single prop is not expressive enough (e.g. JS("v => `${v.first} ${v.last}`")). |
| ... | Additional props to pass to the component. |

### Details

Calls the useSearchParams() hook and injects the result as a prop of the into component. Use the param argument to extract a query parameter by name.

Values are always returned as character vectors so that repeated keys (e.g. "?tag=a&tag=b") are preserved. When injected as "children", vectors are joined with ", "; for custom formatting, use render.

**Reading vs. writing.** The upstream JS hook returns a tuple [searchParams, setSearchParams]. This wrapper splits the two paths:

- into / as — *read-only*. Receives the parsed params (or one param) and ignores the setter.
- render — receives both as (params, setSearchParams), so use this form when you need to update the URL programmatically:

```
useSearchParams(render = JS(
  "(p, set) => <button onClick={() => set({tag:'b'})}>Filter</button>"
))
```

---

useSubmit *useSubmit*

---

### Description

https://api.reactrouter.com/v7/functions/react-router.useSubmit.html

## Usage

```
useSubmit(into = NULL, as = "children", render = NULL, ...)
```

## Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook value as the specified prop. |
| as | Character. The name of the component's prop to inject the hook value into. Defaults to "children". |
| render | Optional JS function (value) => ReactNode used in place of into/as. |
| ... | Additional props to pass to the component. |

## Details

Calls the useSubmit() hook and passes the submit function to render (or injects it as a prop of into). The submit function has signature submit(target, options?) and triggers a form submission (including calling the route's action) without requiring a Form element. Only works inside a data router.

Because the hook returns a function (not a value), the into form is rarely useful here – prefer render = JS(...) so you can call the submit function from inside the rendered element.

## Examples

```
## Not run:
useSubmit(render = JS(
  "submit => <button onClick={() =>
     submit({ intent: 'delete' }, { method: 'post' })
   }>Delete</button>"
))

## End(Not run)
```

---

# useViewTransitionState

*useViewTransitionState*

---

## Description

https://api.reactrouter.com/v7/functions/react-router.useViewTransitionState.html

## Usage

```
useViewTransitionState(
  into = NULL,
  as = "children",
  render = NULL,
  to,
  relative = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| into | A component (HTML tag or **shiny.react**-based element) that will receive the hook value as the specified prop. |
| as | Character. The name of the component's prop to inject the hook value into. Defaults to "children". |
| render | Optional JS function (value) => ReactNode used in place of into/as. |
| to | Character. The destination path being transitioned to. |
| relative | Optional character. Either "route" (default) or "path". |
| ... | Additional props to pass to the component. |

## Details

Calls the useViewTransitionState() hook and injects the boolean result as a prop of the into component. Returns TRUE while a View Transitions API navigation toward to is in progress. Pair with the viewTransition prop on Link/NavLink to drive transition-aware styling.

# Index