SVELTEKIT • REFERENCE

$app/navigation

ON THIS PAGE

```
import {
  afterNavigate,
  beforeNavigate,
  disableScrollHandling,
  goto,
  invalidate,
  invalidateAll,
  onNavigate,
  preloadCode,
  preloadData,
  pushState,
  replaceState
} from '$app/navigation';
```

afterNavigate

A lifecycle function that runs the supplied `callback` when the current component mounts, and also whenever we navigate to a new URL.

`afterNavigate` must be called during a component initialization. It remains active as long as the component is mounted.

```
function afterNavigate(
  callback: (
    navigation: import('@sveltejs/kit').AfterNavigate
  ) => void
): void;
```

A navigation interceptor that triggers before we navigate to a new URL, whether by clicking a link, calling `goto(...)`, or using the browser back/forward controls.

Calling `cancel()` will prevent the navigation from completing. If `navigation.type === 'leave'` — meaning the user is navigating away from the app (or closing the tab) — calling `cancel` will trigger the native browser unload confirmation dialog. In this case, the navigation may or may not be cancelled depending on the user's response.

When a navigation isn't to a SvelteKit-owned route (and therefore controlled by SvelteKit's client-side router), `navigation.to.route.id` will be `null`.

If the navigation will (if not cancelled) cause the document to unload — in other words `'leave'` navigations and `'link'` navigations where `navigation.to.route === null` — `navigation.willUnload` is `true`.

`beforeNavigate` must be called during a component initialization. It remains active as long as the component is mounted.

```
function beforeNavigate(
  callback: (
    navigation: import('@sveltejs/kit').BeforeNavigate
  ) => void
): void;
```

disableScrollHandling

If called when the page is being updated following a navigation (in `onMount` or `afterNavigate` or an action, for example), this disables SvelteKit's built-in scroll handling. This is generally discouraged, since it breaks user expectations.

```
function disableScrollHandling(): void;
```

the promise rejects) to the specified `url`. For external URLs, use `window.location = url` instead of calling `goto(url)`.

```
function goto(
  url: string | URL,
  opts?:
    | {
        replaceState?: boolean | undefined;
        noScroll?: boolean | undefined;
        keepFocus?: boolean | undefined;
        invalidateAll?: boolean | undefined;
        state?: App.PageState | undefined;
      }
    | undefined
): Promise<void>;
```

invalidate

Causes any `load` functions belonging to the currently active page to re-run if they depend on the `url` in question, via `fetch` or `depends`. Returns a `Promise` that resolves when the page is subsequently updated.

If the argument is given as a `string` or `URL`, it must resolve to the same URL that was passed to `fetch` or `depends` (including query parameters). To create a custom identifier, use a string beginning with `[a-z]+:` (e.g. `custom:state`) — this is a valid URL.

The `function` argument can be used define a custom predicate. It receives the full `URL` and causes `load` to rerun if `true` is returned. This can be useful if you want to invalidate based on a pattern instead of a exact match.

```
// Example: Match '/path' regardless of the query parameters
import { invalidate } from '$app/navigation';

invalidate((url) => url.pathname === '/path');
```

invalidateAll

Causes all `load` functions belonging to the currently active page to re-run. Returns a `Promise` that resolves when the page is subsequently updated.

```
function invalidateAll(): Promise<void>;
```

onNavigate

A lifecycle function that runs the supplied `callback` immediately before we navigate to a new URL except during full-page navigations.

If you return a `Promise`, SvelteKit will wait for it to resolve before completing the navigation. This allows you to — for example — use `document.startViewTransition`. Avoid promises that are slow to resolve, since navigation will appear stalled to the user.

If a function (or a `Promise` that resolves to a function) is returned from the callback, it will be called once the DOM has updated.

`onNavigate` must be called during a component initialization. It remains active as long as the component is mounted.

```
function onNavigate(
  callback: (
    navigation: import('@sveltejs/kit').OnNavigate
  ) => MaybePromise<(() => void) | void>
): void;
```

preloadCode

Docs

`src/routes/about/+page.svelte`) or `/blog/*` (to match `src/routes/blog/[slug]/+page.svelte`).

Unlike `preloadData`, this won't call `load` functions. Returns a Promise that resolves when the modules have been imported.

```
function preloadCode(pathname: string): Promise<void>;
```

preloadData

Programmatically preloads the given page, which means

1. ensuring that the code for the page is loaded, and
2. calling the page's load function with the appropriate options.

This is the same behaviour that SvelteKit triggers when the user taps or mouses over an `<a>` element with `data-sveltekit-preload-data`. If the next navigation is to `href`, the values returned from load will be used, making navigation instantaneous. Returns a Promise that resolves with the result of running the new route's `load` functions once the preload is complete.

```
function preloadData(href: string): Promise<
  | {
      type: 'loaded';
      status: number;
      data: Record<string, any>;
    }
  | {
      type: 'redirect';
      location: string;
    }
>;
```

current URL, you can pass `''` as the first argument. Used for shallow routing.

```
function pushState(
  url: string | URL,
  state: App.PageState
): void;
```

replaceState

Programmatically replace the current history entry with the given `$page.state`. To use the current URL, you can pass `''` as the first argument. Used for shallow routing.

```
function replaceState(
  url: string | URL,
  state: App.PageState
): void;
```

Edit this page on GitHub

PREVIOUS
$app/forms

NEXT
$app/paths