SVELTEKIT • ADVANCED

Shallow routing

ON THIS PAGE

As you navigate around a SvelteKit app, you create *history entries*. Clicking the back and forward buttons traverses through this list of entries, re-running any `load` functions and replacing page components as necessary.

Sometimes, it's useful to create history entries *without* navigating. For example, you might want to show a modal dialog that the user can dismiss by navigating back. This is particularly valuable on mobile devices, where swipe gestures are often more natural than interacting directly with the UI. In these cases, a modal that is *not* associated with a history entry can be a source of frustration, as a user may swipe backwards in an attempt to dismiss it and find themselves on the wrong page.

SvelteKit makes this possible with the `pushState` and `replaceState` functions, which allow you to associate state with a history entry without navigating. For example, to implement a history-driven modal:

+page.svelte

```
<script>
  import { pushState } from '$app/navigation';
  import { page } from '$app/stores';
  import Modal from './Modal.svelte';

  function showModal() {
    pushState('', {
      showModal: true
    });
  }
</script>
```

The modal can be dismissed by navigating back (unsetting `$page.state.showModal`) or by interacting with it in a way that causes the `close` callback to run, which will navigate back programmatically.

API

The first argument to `pushState` is the URL, relative to the current URL. To stay on the current URL, use `''`.

The second argument is the new page state, which can be accessed via the page store as `$page.state`. You can make page state type-safe by declaring an `App.PageState` interface (usually in `src/app.d.ts`).

To set page state without creating a new history entry, use `replaceState` instead of `pushState`.

Loading data for a route

When shallow routing, you may want to render another `+page.svelte` inside the current page. For example, clicking on a photo thumbnail could pop up the detail view without navigating to the photo page.

For this to work, you need to load the data that the `+page.svelte` expects. A convenient way to do this is to use `preloadData` inside the `click` handler of an `<a>` element. If the element (or a parent) uses `data-sveltekit-preload-data`, the data will have already been requested, and `preloadData` will reuse that request.

src/routes/photos/+page.svelte

```
<script>
  import { preloadData, pushState, goto } from '$app/navigation';
```

```
  let { data } = $props();
</script>

{#each data.thumbnails as thumbnail}
  <a
    href="/photos/{thumbnail.id}"
    on:click={async (e) => {
      if (innerWidth < 640        // bail if the screen is too small
        || e.shiftKey             // or the link is opened in a new window
        || e.metaKey || e.ctrlKey // or a new tab (mac: metaKey, win/linux: ctrlKey)
        // should also consider clicking with a mouse scroll wheel
      ) return;

      // prevent navigation
      e.preventDefault();

      const { href } = e.currentTarget;

      // run `load` functions (or rather, get the result of the `load` functions
      // that are already running because of `data-sveltekit-preload-data`)
      const result = await preloadData(href);

      if (result.type === 'loaded' && result.status === 200) {
        pushState(href, { selected: result.data });
      } else {
        // something bad happened! try navigating
        goto(href);
      }
    }}
  >
    <img alt={thumbnail.alt} src={thumbnail.src} />
  </a>
{/each}

{#if $page.state.selected}
  <Modal on:close={() => history.back()}>
    <!-- pass page data to the +page.svelte component,
         just like SvelteKit would on navigation -->
    <PhotoPage data={$page.state.selected} />
  </Modal>
{/if}
```

During server-side rendering, `$page.state` is always an empty object. The same is true for the first page the user lands on — if the user reloads the page (or returns from another document), state will *not* be applied until they navigate.

Shallow routing is a feature that requires JavaScript to work. Be mindful when using it and try to think of sensible fallback behavior in case JavaScript isn't available.

Edit this page on GitHub

PREVIOUS
Snapshots

NEXT
Packaging