SVELTEKIT • APPENDIX

Migrating from Sapper

ON THIS PAGE

SvelteKit is the successor to Sapper and shares many elements of its design.

If you have an existing Sapper app that you plan to migrate to SvelteKit, there are a number of changes you will need to make. You may find it helpful to view some examples while migrating.

package.json

type: "module"

Add `"type": "module"` to your `package.json`. You can do this step separately from the rest as part of an incremental migration if you are using Sapper 0.29.3 or newer.

dependencies

Remove `polka` or `express`, if you're using one of those, and any middleware such as `sirv` or `compression`.

devDependencies

Remove `sapper` from your `devDependencies` and replace it with `@sveltejs/kit` and whichever adapter you plan to use (see next section).

`sapper build` should become `vite build` using the Node adapter

`sapper export` should become `vite build` using the static adapter

`sapper dev` should become `vite dev`

`node __sapper__/build` should become `node build`

Project files

The bulk of your app, in `src/routes`, can be left where it is, but several project files will need to be moved or updated.

Configuration

Your `webpack.config.js` or `rollup.config.js` should be replaced with a `svelte.config.js`, as documented here. Svelte preprocessor options should be moved to `config.preprocess`.

You will need to add an adapter. `sapper build` is roughly equivalent to adapter-node while `sapper export` is roughly equivalent to adapter-static, though you might prefer to use an adapter designed for the platform you're deploying to.

If you were using plugins for filetypes that are not automatically handled by Vite, you will need to find Vite equivalents and add them to the Vite config.

src/client.js

This file has no equivalent in SvelteKit. Any custom logic (beyond `sapper.start(...)`) should be expressed in your `+layout.svelte` file, inside an `onMount` callback.

src/server.js

Most imports from `@sapper/service-worker` have equivalents in `$service-worker`:

- `files` is unchanged
- `routes` has been removed
- `shell` is now `build`
- `timestamp` is now `version`

src/template.html

The `src/template.html` file should be renamed `src/app.html`.

Remove `%sapper.base%`, `%sapper.scripts%` and `%sapper.styles%`. Replace `%sapper.head%` with `%sveltekit.head%` and `%sapper.html%` with `%sveltekit.body%`. The `<div id="sapper">` is no longer necessary.

src/node_modules

A common pattern in Sapper apps is to put your internal library in a directory inside `src/node_modules`. This doesn't work with Vite, so we use `src/lib` instead.

Pages and layouts

Renamed files

Routes now are made up of the folder name exclusively to remove ambiguity, the folder names leading up to a `+page.svelte` correspond to the route. See the routing docs for an overview. The following shows a old/new comparison:

Old	New
routes/about/index.svelte	routes/about/+page.svelte

`+error.svelte`. Any `_layout.svelte` files should likewise be renamed `+layout.svelte`. Any other files are ignored.

Imports

The `goto`, `prefetch` and `prefetchRoutes` imports from `@sapper/app` should be replaced with `goto`, `preloadData` and `preloadCode` imports respectively from `$app/navigation`.

The `stores` import from `@sapper/app` should be replaced — see the Stores section below.

Any files you previously imported from directories in `src/node_modules` will need to be replaced with `$lib` imports.

Preload

As before, pages and layouts can export a function that allows data to be loaded before rendering takes place.

This function has been renamed from `preload` to `load`, it now lives in a `+page.js` (or `+layout.js`) next to its `+page.svelte` (or `+layout.svelte`), and its API has changed. Instead of two arguments — `page` and `session` — there is a single `event` argument.

There is no more `this` object, and consequently no `this.fetch`, `this.error` or `this.redirect`. Instead, you can get `fetch` from the input methods, and both `error` and `redirect` are now thrown.

Stores

In Sapper, you would get references to provided stores like so:

```
import { stores } from '@sapper/app';
const { preloading, page, session } = stores();
```

The `stores` still exists [illegible] has been replaced with a [illegible] stores that

You access them differently in SvelteKit. `stores` is now `getStores`, but in most cases it is unnecessary since you can import `navigating`, and `page` directly from `$app/stores`.

Routing

Regex routes are no longer supported. Instead, use advanced route matching.

Segments

Previously, layout components received a `segment` prop indicating the child segment. This has been removed; you should use the more flexible `$page.url.pathname` value to derive the segment you're interested in.

URLs

In Sapper, all relative URLs were resolved against the base URL — usually `/`, unless the `basepath` option was used — rather than against the current page.

This caused problems and is no longer the case in SvelteKit. Instead, relative URLs are resolved against the current page (or the destination page, for `fetch` URLs in `load` functions) instead. In most cases, it's easier to use root-relative (i.e. starts with `/`) URLs, since their meaning is not context-dependent.

<a> attributes

`sapper:prefetch` is now `data-sveltekit-preload-data`

`sapper:noscroll` is now `data-sveltekit-noscroll`

Endpoints

on a Node server, but could equally be running on a serverless platform or in a Cloudflare Worker. For that reason, you no longer interact directly with `req` and `res`. Your endpoints will need to be updated to match the new signature.

To support this environment-agnostic behavior, `fetch` is now available in the global context, so you don't need to import `node-fetch`, `cross-fetch`, or similar server-side fetch implementations in order to use it.

Integrations

See integrations for detailed information about integrations.

HTML minifier

Sapper includes `html-minifier` by default. SvelteKit does not include this, but you can add it as a prod dependency and then use it through a hook:

```
import { minify } from 'html-minifier';
import { building } from '$app/environment';

const minification_options = {
  collapseBooleanAttributes: true,
  collapseWhitespace: true,
  conservativeCollapse: true,
  decodeEntities: true,
  html5: true,
  ignoreCustomComments: [/^#/],
  minifyCSS: true,
  minifyJS: false,
  removeAttributeQuotes: true,
  removeComments: false, // some hydration code needs comments, so leave them in
  removeOptionalTags: true,
  removeRedundantAttributes: true,
  removeScriptTypeAttributes: true,
  removeStyleLinkTypeAttributes: true,
  sortAttributes: true,
```

```
export async function handle({ event, resolve }) {
  let page = '';

  return resolve(event, {
    transformPageChunk: ({ html, done }) => {
      page += html;
      if (done) {
        return building ? minify(page, minification_options) : page;
      }
    }
  });
}
```

Note that `prerendering` is `false` when using `vite preview` to test the production build of the site, so to verify the results of minifying, you'll need to inspect the built HTML files directly.

Edit this page on GitHub

PREVIOUS
Migrating to SvelteKit v2

NEXT
Additional resources