If you *didn't* attend Svelte Summit, you can catch up on the Svelte Society YouTube page

If you attended Svelte Summit last month you may have seen my talk, Futuristic Web Development, in which I finally tackled one of the most frequently asked questions about Svelte: when will Sapper reach version 1.0?

The answer: never.

This was slightly tongue-in-cheek — as the talk explains, it's really more of a rewrite of Sapper coupled with a rebrand — but it raised a lot of new questions from the community, and it's time we offered a bit more clarity on what you can expect from Sapper's successor, SvelteKit.

'Futuristic Web Development' from Svelte Summit

What's Sapper?

Sapper is an *app framework* (or 'metaframework') built on top of Svelte (which is a *component* framework). Its job is to make it easy to build Svelte apps with all the modern best practices like server-side rendering (SSR) and code-splitting, and to provide a project structure that makes development productive and fun. It uses *filesystem-based routing* (as popularised by Next and adopted by many other frameworks, albeit with some enhancements) — your project's file structure mirrors the structure of the app itself.

While the Svelte homepage and documentation encourages you to degit the sveltejs/template repo to start building an app, Sapper has long been our recommended way to build apps; this very blog post is (at the time of writing!) rendered with Sapper.

Why are we migrating to something new?

Firstly, the distinction between sveltejs/template and sveltejs/sapper-template is confusing, particularly to newcomers to Svelte. Having a single recommended way to start building apps with Svelte will bring enormous benefits: we simplify onboarding, reduce the maintenance and support burden, and can potentially begin to explore the new possibilities that are unlocked by having a predictable project structure. (This last part is deliberately vague because it will take time to fully understand what those possibilities are.)

Aside from all that, we've been tempted by the thought of rewriting Sapper for a while. This is partly because the codebase has become a little unkempt over the years (Sapper started in 2017), but mostly because the web has changed a lot recently, and it's time to rethink some of our foundational assumptions.

How is this new thing different?

The first of those foundational assumptions is that you need to use a module bundler like webpack or Rollup to build apps. These tools trace the dependency graph of your application, analysing and transforming code along the way (turning Svelte components to JS modules, for example), in order to create bundles of code that can run anywhere. As the original creator of Rollup, I can attest that it is a surprisingly complex problem with fiendish edge cases.

You certainly needed a bundler several years ago, because browsers didn't natively support the `import` keyword, but it's much less true today. Right now, we're seeing the rise of the *unbundled development* workflow, which is radically simpler: instead of eagerly bundling your app, a dev server can serve modules (converted to JavaScript, if necessary) *on-demand*, meaning startup is essentially instantaneous however large your app becomes.

Snowpack is at the vanguard of this movement, and it's what powers SvelteKit. It's astonishingly fast, and has a beautiful development experience (hot module reloading, error overlays and so on), and we've been working closely with the Snowpack team on features like SSR. The hot module reloading is particularly revelatory if you're used to using Sapper with Rollup (which has never had first-class HMR support owing to its architecture, which prioritises the most efficient output).

That's not to say we're abandoning bundlers altogether. It's still essential to optimise your app for production, and SvelteKit uses Rollup to make your apps as fast and lean as they possibly can be (which includes things like extracting styles into static `.css` files).

The other foundational assumption is that a server-rendered app needs, well, a server. Sapper effectively has two modes — `sapper build`, which creates a standalone app that has to run on a Node server, and `sapper export` which bakes your app out as a collection of static files suitable for hosting on services like GitHub Pages.

Static files can go pretty much anywhere, but running a Node server (and monitoring/scaling it etc) is less straightforward. Nowadays we're witnessing a shift towards *serverless platforms*, in which you as the app author don't need to think about the server your code is running on, with all the attendant complexity. You can get Sapper apps running on serverless platforms, thanks to things like vercel-sapper, but it's certainly not what you'd call idiomatic.

It'll still be possible to create both Node apps and fully pre-rendered (aka exported) sites

SvelteKit fully embraces the serverless paradigm, and will launch with support for all the major serverless providers, with an 'adapter' API for targeting any platforms that we don't officially cater to. In addition, we'll be able to do partial pre-rendering, which means that static pages can be generated at build time but dynamic ones get rendered on-demand.

When can I start using it?

If you're feeling brave, you can start right now:

```
npm init svelte@next
```

This will scaffold a new project and install the `@sveltejs/kit` CLI, which provides the tools for developing and building an app.

We don't recommend it though! There are no docs, and we won't be able to offer any form of support. It's also likely to break often.

The work is being done in a private monorepo while we're still in exploration mode. Our plan is to get a public beta ready and announce it here once we've closed a few issues — the repo itself will remain private at that time, but we'll create a place to collect feedback from the YOLO crowd. After that, we'll work towards a 1.0 release which will involve opening the repo up.

I'm not going to make any firm promises about timings, because I don't like to break promises. But I *think* we're talking about weeks rather than months.

What if I don't want to use SvelteKit?

You won't have to — it will always be possible to use Svelte as a standalone package or via a bundler integration like rollup-plugin-svelte. We think it's essential that you can bend Svelte to fit your workflow, however esoteric, and use third-party app frameworks like Elder.js, Routify, Plenti, Crown, JungleJS and others.

TypeScript?

Don't worry, we won't launch without full TypeScript support.

How can I migrate my existing Sapper apps?

For the most part, it should be relatively straightforward to migrate a Sapper codebase.

There are some unavoidable changes (being able to run on serverless platforms means we need to replace custom `server.js` files and `(req, res) => {...}` functions with more portable equivalents), and we're taking the opportunity to fix a few design flaws, but on the whole a SvelteKit app will feel very familiar to Sapper users.

Detailed migration guides will accompany the 1.0 launch.

How can I contribute?

Keep your eyes peeled for announcements about when we'll launch the public beta and open up the repo. (Also, blog post TODO but I would be remiss if I didn't mention that we now have an OpenCollective where you can contribute financially to the project if it's been valuable to you. Many, many thanks to those of you who already have.)

Where can I learn more?

Follow @sveltejs and @SvelteSociety on Twitter, and visit svelte.dev/chat. You should also subscribe to Svelte Radio, where Kevin and his co-hosts will grill me about this project on an upcoming episode (and between now and next week when we record it, reply to this Twitter thread with your additional questions).