It's been by far the most requested feature for a while, and it's finally here: Svelte officially supports TypeScript.

We think it'll give you a much nicer development experience — one that also scales beautifully to larger Svelte code bases — regardless of whether you use TypeScript or JavaScript.

 Screenshot of TypeScript in Svelte

Image of TypeScript + Svelte in VS Code (theme is Kary Pro.)

Try it now

You can start a new Svelte TypeScript project using the normal template and by running `node scripts/setupTypeScript.js` before you do anything else:

```
npx degit sveltejs/template svelte-typescript-app
cd svelte-typescript-app
node scripts/setupTypeScript.js
```

If you're a VS Code user, make sure you're using the (new) official extension, which replaces the popular extension by James Birtles. Later in this blog post, we'll detail the individual steps involved in using TypeScript in an existing Svelte project.

What does it mean to support TypeScript in Svelte?

TypeScript support in Svelte has been possible for a long time, but you had to mix a lot of disparate tools together and each project ran independently. Today, nearly all of these tools live under the Svelte organization and are maintained by a set of people who take responsibility over the whole pipeline and have common goals.

A week before COVID was declared a pandemic, I pitched a consolidation of the best Svelte tools and ideas from similar dev-ecosystems and provided a set of steps to get first class TypeScript support. Since then, many people have pitched in and written the code to get us there.

When we say that Svelte now supports TypeScript, we mean a few different things:

- You can use TypeScript inside your `<script>` blocks — just add the `lang="ts"` attribute
- Components with TypeScript can be type-checked with the `svelte-check` command
- You get autocompletion hints and type-checking as you're writing components, even in expressions inside markup
- TypeScript files understand the Svelte component API — no more red squiggles when you import a `.svelte` file into a `.ts` module

How does it work?

To understand the two main parts of TypeScript support, we'll compare it to the technique TypeScript uses to provide dev tools. There is a compiler `tsc` which you run on the command-line to convert `*.ts` to `*.js`, then there is a `TSServer` which is a node API that responds to requests from text editors. The `TSServer` is what provides all the JavaScript and TypeScript realtime introspection for editors while coding, and it has most of the compiler's code inside it.

For Svelte, we have the Svelte compiler, and now we have the `svelte-language-server` which responds to text editor calls via the Language Server Protocol standard. First class TypeScript support means that *both* of these two systems do a good job of handling TypeScript code.

The Svelte compiler support for TypeScript is handled by Christian Kaisermann's `svelte-preprocess` which is now an official Svelte project.

For the editor level, we took inspiration from Pine's work in the Vue ecosystem via Vetur. Vetur provides an LSP, a VS Code extension and a CLI. Svelte now also has an LSP, a VS Code extension and a CLI.

`*.svelte` Introspection

For the official Svelte VS Code extension, we built off the foundations which James Birtles has created in `UnwrittenFun/svelte-vscode` and `UnwrittenFun/svelte-language-server`.

Simon Holthausen and Lyu, Wei-Da have done great work improving the JavaScript and TypeScript introspection, including integrating @halfnelson's svelte2tsx which powers understanding the props on components in your codebase.

Adding TypeScript to an existing project

Before getting started, add the dependencies:

```
npm install --save-dev @tsconfig/svelte typescript svelte-preprocess svelte-check
```

1. Compiling TypeScript

You first need to set up `svelte-preprocess`, which passes the contents of your `<script lang="ts">` blocks through the TypeScript compiler.

In a Rollup project, that would look like this — note that we also need to install `@rollup/plugin-typescript` so that Rollup can handle `.ts` files:

```
+ import autoPreprocess from 'svelte-preprocess';
+ import typescript from '@rollup/plugin-typescript';

export default {
  ...,
  plugins: [
    svelte({
+       preprocess: autoPreprocess()
    }),
+   typescript({ sourceMap: !production })
  ]
}
```

Full instructions for other environments here.

To configure TypeScript, you will need to create a `tsconfig.json` in the root of your project:

```
{
  "extends": "@tsconfig/svelte/tsconfig.json",

  "include": ["src/**/*", "src/node_modules"],
```

```
  "exclude": ["node_modules/*", "__sapper__/*", "public/*"],
}
```

Your `include` / `exclude` may differ per project — these are defaults that should work across most Svelte projects.

2. Editor Support

Any editor using an LSP can be supported. The VS Code extension has been our primary focus, but there is work in progress on Atom, and Vim via coc-svelte has been updated with the latest LSP.

These editor extensions will improve your coding experience even if you only use JavaScript. The editor won't offer errors, but it will offer inference and refactoring tools. You can add `// @ts-check` to the top of a `<script>` tag using JavaScript to get better error messages with no infra changes.

To switch a `<script>` to use TypeScript, use `<script lang="ts">` and that should be it. Hopefully you won't be seeing an ocean of red squiggles.

3. CI Checks

Having red squiggles is great, well, kinda. On the long run though, you want to be able to verify that there are no errors in your code. To verify your project is error free, you can use the CLI tool `svelte-check`. It acts like an editor asking for errors against all of your `.svelte` files.

You can add the dependency to your project and then add it to CI.

```
❯ npx svelte-check

Loading svelte-check in workspace: /Users/ortatherox/dev/svelte/example-app
Getting Svelte diagnostics...
====================================

/Users/ortatherox/dev/svelte/example-app/src/App.svelte:3:2
Error: Type '123' is not assignable to type 'string'. (ts)

====================================
svelte-check found 1 error
error Command failed with exit code 1.
```

What about TypeScript in Sapper projects?

TypeScript support was added to Sapper in 0.28, so if you're using an older version be sure to upgrade.

How can I contribute?

We're so glad you asked. The work is happening in the sveltejs/language-tools repo and in the #language-tools channel in the Svelte Discord. If you'd like to report issues, submit fixes, or help out with extensions for new editors and so on, that's where you can find us. See you there!