Frequently asked questions

ON THIS PAGE

I'm new to Svelte. Where should I start?

We think the best way to get started is playing through the interactive tutorial. Each step there is mainly focused on one specific aspect and is easy to follow. You'll be editing and running real Svelte components right in your browser.

Five to ten minutes should be enough to get you up and running. An hour and a half should get you through the entire tutorial.

Where can I get support?

If your question is about certain syntax, the reference docs are a good place to start.

Stack Overflow is a popular forum to ask code-level questions or if you're stuck with a specific error. Read through the existing questions tagged with Svelte or ask your own!

There are online forums and chats which are a great place for discussion about best practices, application architecture or just to get to know fellow Svelte users. Our Discord or the Reddit channel are examples of that. If you have an answerable code-level question, Stack Overflow is usually a better fit.

Are there any third-party resources?

Svelte Society maintains a list of books and videos.

There is an official VS Code extension for Svelte.

Is there a tool to automatically format my .svelte files?

You can use prettier with the prettier-plugin-svelte plugin.

How do I document my components?

In editors which use the Svelte Language Server you can document Components, functions and exports using specially formatted comments.

~~~svelte
<script>
  /** What should we call the user? */
  export let name = 'world';
</script>

<!--
@component
Here's some documentation for this component.
It will show up on hover.

- You can use markdown here.
- You can also use code blocks here.
- Usage:
  ```tsx
  <main name="Arethra">
  ~~~
-->
<main>
  <h1>
    Hello, {name}
  </h1>
</main>
```
```

Does Svelte scale?

There will be a blog post about this eventually, but in the meantime, check out this issue.

Is there a UI component library?

There are several UI component libraries as well as standalone components. Find them under the design systems section of the components page on the Svelte Society website.

How do I test Svelte apps?

How your application is structured and where logic is defined will determine the best way to ensure it is properly tested. It is important to note that not all logic belongs within a component - this includes concerns such as data transformation, cross-component state management, and logging, among others. Remember that the Svelte library has its own test suite, so you do not need to write tests to validate implementation details provided by Svelte.

A Svelte application will typically have three different types of tests: Unit, Component, and End-to-End (E2E).

Unit Tests: Focus on testing business logic in isolation. Often this is validating individual functions and edge cases. By minimizing the surface area of these tests they can be kept lean and fast, and by extracting as much logic as possible from your Svelte components more of your application can be covered using them. When creating a new SvelteKit project, you will be asked whether you would like to setup Vitest for unit testing. There are a number of other test runners that could be used as well.

Component Tests: Validating that a Svelte component mounts and interacts as expected throughout its lifecycle requires a tool that provides a Document Object Model (DOM).

capabilities provided by a Svelte component. Tools for component testing range from an in-memory implementation like jsdom paired with a test runner like Vitest to solutions that leverage an actual browser to provide a visual testing capability such as Playwright or Cypress.

End-to-End Tests: To ensure your users are able to interact with your application it is necessary to test it as a whole in a manner as close to production as possible. This is done by writing end-to-end (E2E) tests which load and interact with a deployed version of your application in order to simulate how the user will interact with your application. When creating a new SvelteKit project, you will be asked whether you would like to setup Playwright for end-to-end testing. There are many other E2E test libraries available for use as well.

Some resources for getting started with testing:

- Svelte Testing Library
- Svelte Component Testing in Cypress
- Example using vitest
- Example using uvu test runner with JSDOM
- Test Svelte components using Vitest & Playwright
- Component testing with WebdriverIO

Is there a router?

The official routing library is SvelteKit. SvelteKit provides a filesystem router, server-side rendering (SSR), and hot module reloading (HMR) in one easy-to-use package. It shares similarities with Next.js for React.

However, you can use any router library. A lot of people use page.js. There's also navaid, which is very similar. And universal-router, which is isomorphic with child routes, but without built-in history support.

If you need hash-based routing on the client side, check out svelte-spa-router or abstract-state-router.

Routify is another filesystem-based router, similar to SvelteKit's router. Version 3 supports Svelte's native SSR.

You can see a community-maintained list of routers on sveltesociety.dev.

Can I tell Svelte not to remove my unused styles?

No. Svelte removes the styles from the component and warns you about them in order to prevent issues that would otherwise arise.

Svelte's component style scoping works by generating a class unique to the given component, adding it to the relevant elements in the component that are under Svelte's control, and then adding it to each of the selectors in that component's styles. When the compiler can't see what elements a style selector applies to, there would be two bad options for keeping it:

- If it keeps the selector and adds the scoping class to it, the selector will likely not match the expected elements in the component, and they definitely won't if they were created by a child component or `{@html ...}`.
- If it keeps the selector without adding the scoping class to it, the given style will become a global style, affecting your entire page.

If you need to style something that Svelte can't identify at compile time, you will need to explicitly opt into global styles by using `:global(...)`. But also keep in mind that you can wrap `:global(...)` around only part of a selector. `.foo :global(.bar) { ... }` will style any `.bar` elements that appear within the component's `.foo` elements. As long as there's some parent element in the current component to start from, partially global selectors like this will almost always be able to get you what you want.

New features aren't being added to it, and bugs will probably only be fixed if they are extremely nasty or present some sort of security vulnerability.

The documentation is still available here.

How do I do hot module reloading?

We recommend using SvelteKit, which supports HMR out of the box and is built on top of Vite and svelte-hmr. There are also community plugins for rollup and webpack.

Edit this page on GitHub

PREVIOUS
Svelte 5 migration guide

NEXT
svelte