Contributing to Svelte

Svelte is a new way to build web applications. It's a compiler that takes your declarative components and converts them into efficient JavaScript that surgically updates the DOM.

The Open Source Guides website has a collection of resources for individuals, communities, and companies. These resources help people who want to learn how to run and contribute to open source projects. Contributors and people new to open source alike will find the following guides especially useful:

- How to Contribute to Open Source
- Building Welcoming Communities

Get involved

There are many ways to contribute to Svelte, and many of them do not involve writing any code. Here's a few ideas to get started:

- Simply start using Svelte. Go through the Getting Started guide. Does everything work as expected? If not, we're always looking for improvements. Let us know by opening an issue.
- Look through the open issues. A good starting point would be issues tagged good first issue. Provide workarounds, ask for clarification, or suggest labels. Help triage issues.
- If you find an issue you would like to fix, open a pull request.
- Read through our tutorials. If you find anything that is confusing or can be improved, you can make edits by clicking "Edit this page" at the bottom left of the tutorial page.
- Take a look at the features requested by others in the community and consider opening a pull request if you see something you want to work on.

Contributions are very welcome. If you think you need help planning your contribution, please ping us on Discord at svelte.dev/chat and let us know you are looking for a bit of help.

Triaging issues and pull requests

One great way you can contribute to the project without writing any code is to help triage issues and pull requests as they come in.

- Ask for more information if you believe the issue does not provide all the details required to solve it.
- Flag issues that are stale or that should be closed.
- Ask for test plans and review code.

Our process

RFCs

If you'd like to propose an implementation for a large new feature or change then please create an RFC to discuss it up front.

Roadmap

When deciding where to contribute, you may wish to take a look at our roadmap. The Svelte team generally works on a single major effort at a time. This has a couple benefits for us as maintainers. First, it allows us to focus and make noticeable progress in an area being proactive rather than reactive. Secondly, it allows us to handle related issues and PRs together. By batching issues and PRs together we're able to ensure implementations and fixes holistically address the set of problems and use cases encountered by our users.

Maintainer meetings

The maintainers meet on the final Saturday of each month. While these meetings are not open publicly, we will report back by leaving a comment on each issue discussed. We will generally discuss items aligning with our roadmap, but major PRs needing discussion amongst the maintainers can be added to the agenda for the monthly maintainer's meeting. However, we typically are only able to get to a couple of items that are not aligned with our current priority.

Prioritization

We do our best to review PRs and RFCs as they are sent, but it is difficult to keep up. We welcome help in reviewing PRs, RFCs, and issues. If an item aligns with the current priority on our roadmap, it is more likely to be reviewed quickly. PRs to the most important and active ones repositories get reviewed more quickly while PRs to smaller inactive repos may sit for a bit before we periodically come by and review the pending PRs in a batch.

Bugs

We use GitHub issues for our public bugs. If you would like to report a problem, take a look around and see if someone already opened an issue about it. If you are certain this is a new unreported bug, you can submit a bug report.

If you have questions about using Svelte, contact us on Discord at svelte.dev/chat, and we will do our best to answer your questions.

If you see anything you'd like to be implemented, create a feature request issue

Reporting new issues

When opening a new issue, always make sure to fill out the issue template. **This step is very important!** Not doing so may result in your issue not being managed in a timely fashion. Don't take this personally if this happens, and feel free to open a new issue once you've gathered all the information required by the template.

- **One issue, one bug:** Please report a single bug per issue.
- **Provide reproduction steps:** List all the steps necessary to reproduce the issue. The person reading your bug report should be able to follow these steps to reproduce your issue with minimal effort. If possible, use the REPL to create your reproduction.

Pull requests

> HEADS UP: Svelte 5 will likely change a lot on the compiler. For that reason, please don't open PRs that are large in scope, touch more than a couple of files etc. In other words, bug fixes are fine, but big feature PRs will likely not be merged.

Proposing a change

If you would like to request a new feature or enhancement but are not yet thinking about opening a pull request, you can also file an issue with feature template.

If you're only fixing a bug, it's fine to submit a pull request right away, but we still recommend that you file an issue detailing what you're fixing. This is helpful in case we don't accept that specific fix but want to keep track of the issue.

Small pull requests are much easier to review and more likely to get merged.

Installation

Ensure you have pnpm installed. After cloning the repository, run `pnpm install`.

Developing

To build the UMD version of `svelte/compiler` (this is only necessary for CommonJS consumers, or in-browser use), run `pnpm build` inside `packages/svelte`. To rebuild whenever source files change, run `pnpm dev`.

Creating a branch

Fork the repository and create your branch from `main`. If you've never sent a GitHub pull request before, you can learn how from this free video series.

Testing

A good test plan has the exact commands you ran and their output, provides screenshots or videos if the pull request changes UI.

- If you've changed APIs, update the documentation.

Writing tests

All tests are located in `/test` folder.

Test samples are kept in `/test/xxx/samples` folder.

Running tests

> PREREQUISITE: Install chromium via playwright by running `pnpm playwright install chromium`

1. To run test, run `pnpm test` .
2. To run a particular test suite, use `pnpm test <suite-name>` , for example:

   ```
   pnpm test validator
   ```

3. To filter tests *within* a test suite, use `pnpm test <suite-name> -- -t <test-name>` , for example:

   ```
   pnpm test validator -- -t a11y-alt-text
   ```

   (You can also do `FILTER=<test-name> pnpm test <suite-name>` which removes other tests rather than simply skipping them — this will result in faster and more compact test results, but it's non-idiomatic. Choose your fighter.)

Updating `.expected` files

1. Tests suites like `snapshot` and `parser` assert that the generated output matches the existing snapshot.
2. To update these snapshots, run `UPDATE_SNAPSHOTS=true pnpm test` .

Typechecking

To typecheck the codebase, run `pnpm check` inside `packages/svelte` . To typecheck in watch mode, run `pnpm check:watch` .

Style guide

Eslint will catch most styling issues that may exist in your code. You can check the status of your code styling by simply running `pnpm lint`.

Code conventions

- `snake_case` for internal variable names and methods.
- `camelCase` for public variable names and methods.

Generating types

Types are auto-generated from the source, but the result is checked in to ensure no accidental changes slip through. Run `pnpm generate:types` to regenerate the types.

Sending your pull request

Please make sure the following is done when submitting a pull request:

1. Describe your **test plan** in your pull request description. Make sure to test your changes.
2. Make sure your code lints (`pnpm lint`).
3. Make sure your tests pass (`pnpm test`).

All pull requests should be opened against the `main` branch. Make sure the PR does only one thing, otherwise please split it. If this change should contribute to a version bump, run `npx changeset` at the root of the repository after a code change and select the appropriate packages.

Breaking changes

When adding a new breaking change, follow this template in your pull request:

```
### New breaking change here

- **Who does this affect**:
- **How to migrate**:
- **Why make this breaking change**:
- **Severity (number of people affected x effort)**:
```

Reviewing pull requests

If you'd like to manually test a pull request in another pnpm project, you can do so by running `pnpm add -D "github:sveltejs/svelte#path:packages/svelte&branch-name"` in that project.

License

By contributing to Svelte, you agree that your contributions will be licensed under its MIT license.

Questions

Feel free to ask in #contributing on Discord if you have questions about our process, how to proceed, etc.