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 chapter" 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.
- Suggest labels that can help categorize issues.
- Flag issues that are stale or that should be closed.
- Ask for test plans and review code.

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.

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.

Installation

1. Ensure you have npm installed.
2. After cloning the repository, run `npm install` in the root of the repository.
3. To start a development server, run `npm run dev`.

Pull requests

Your first pull request

So you have decided to contribute code back to upstream by opening a pull request. You've invested a good chunk of time, and we appreciate it. We will do our best to work with you and get the PR looked at.

Working on your first Pull Request? You can learn how from this free video series:

How to Contribute to an Open Source Project on GitHub

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.

Sending a pull request

Small pull requests are much easier to review and more likely to get merged. Make sure the PR does only one thing, otherwise please split it.

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

1. Fork the repository and create your branch from `master`.
2. Describe your **test plan** in your pull request description. Make sure to test your changes.
3. Make sure your code lints (`npm run lint`).
4. Make sure your tests pass (`npm run test`).

All pull requests should be opened against the `master` branch.

Test plan 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

1. To run test, run `npm run test`.
2. To run test for a specific feature, you can use the `-g` (aka `--grep`) option. For example, to only run test involving transitions, run `npm run test -- -g transition`.

Running solo test

1. To run only one test, rename the test sample folder to end with `.solo`. For example, to run the `test/js/samples/action` only, rename it to `test/js/samples/action.solo`.
2. To run only one test suite, rename the test suite folder to end with `.solo`. For example, to run the `test/js` test suite only, rename it to `test/js.solo`.

3. Remember to rename the test folder back. The CI will fail if there's a solo test.

Updating `.expected` files

1. Tests suites like `css`, `js`, `server-side-rendering` asserts that the generated output has to match the content in the `.expected` file. For example, in the `js` test suites, the generated js code is compared against the content in `expected.js`.
2. To update the content of the `.expected` file, run the test with `--update` flag. (`npm run test --update`)

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)**:
```

What happens next?

The core Svelte team will be monitoring for pull requests. Do help us by making your pull request easy to review by following the guidelines above.

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 `npm run lint`.

Code conventions

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

License

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