# Contributing to Linaria

## [Code of Conduct](/CODE\_OF\_CONDUCT.md)

We want this community to be friendly and respectful to each other. Please read [the full text](/CODE\_OF\_CONDUCT.md) so that you can understand what actions will and will not be tolerated.

## Our Development Process

The core team works directly on GitHub and all work is public.

### Development workflow

> \*\*Working on your first pull request?\*\* You can learn how from this \*free\* series: [How to Contribute to an Open Source Project on GitHub](https://egghead.io/series/how-to-contribute-to-an-open-source-project-on-github).

1. Fork the repo and create your branch from `master` (a guide on [how to fork a repository](https://help.github.com/articles/fork-a-repo/)).

1. Linaria uses [Yarn](https://yarnpkg.com/en/) for running development scripts. If you haven't already done so, please [install yarn](https://yarnpkg.com/en/docs/install).

1. Run `yarn bootstrap` to setup the development environment.

1. When working on the code, you can use the watch mode to watch and compiles changed files:

```sh

yarn watch

```

1. If you've added code that should be tested, add tests.

1. You can test the changes you are making in our website located under the `website/` folder in the repo.

1. If you've changed APIs, remember to update the documentation under `docs/`.

### Working on the website

If you are making changes to the website, test the website folder and run the server to check if your changes are being displayed accurately. To run the development server with live reloading, run:

```sh

yarn website start

```

### Commit message convention

We follow the [conventional commits specification](https://www.conventionalcommits.org/en) for our commit messages:

\* `fix`: bug fixes, e.g. fix wrong classname generated by babel plugin.

\* `feat`: new features, e.g. add support for a config file.

\* `refactor`: code refactor, e.g. split the babel plugin into multiple visitors.

\* `docs`: changes into documentation, e.g. add usage example for styled tag.

\* `test`: adding or updating tests, eg unit, snapshot testing.

\* `chore`: tooling changes, e.g. change the eslint config.

\* `BREAKING CHANGE`: for changes that break existing usage, e.g. change the API or behaviour.

Our pre-commit hooks verify that your commit message matches this format when committing. If you are working on some in progress commits and don't care about the pre-commit hook, pass the `--no-verify` (`-n`) flag to git when committing (e.g. `git commit -n -m "wip"`).

### Linting and tests

We use `flow` for type checking, `eslint` with `prettier` for linting and formatting the code, and `jest` for testing. Our pre-commit hooks verify that the linter and tests pass when commiting. You can also run the following commands manually:

\* `yarn flow`: run flow on all files.

\* `yarn typescript`: run tests for typescript definitions.

\* `yarn lint`: lint files with eslint and prettier.

\* `yarn test`: run unit tests with jest.

\* `yarn test:integration`: run integration tests with jest and puppeteer.

### Sending a pull request

When you're sending a pull request:

\* Prefer small pull requests focused on one change.

\* Verify that `flow`, `eslint` and all tests are passing.

\* Preview the documentation to make sure it looks good.

\* Follow the pull request template when opening a pull request.

## How to try a development build of Linaria in a project

To link `linaria` on the command line to local copy in a development build:

```sh

cd /path/to/your/linaria\_clone/

yarn

yarn link

cd /path/to/test\_project/

yarn link linaria

```

To unlink it:

```sh

yarn unlink linaria

```

### Debugging and deep dive into babel plugin

To understand what is actually going in under the hood of Linaria we recommend to first read the top-level [How it works](/docs/HOW\_IT\_WORKS.md).

Then you can run Linaria in debug mode for a specific file from you project and see step-by-step logs informing what is happening.

```sh

DEBUG=linaria\* LINARIA\_LOG=debug npx babel -d dist PATH\_TO\_COMPONENT

```

> In example we use `npx` to run babel, but you can also put this script into package.json with removed `npx` to run it.

You will see how the babel part of Linaria work, result JS file will be stored in `dist` directory. You will not see the resulting CSS file, because this is the role of next layer - bundlers (CSS content is stored in babel metadata to be consumed by bundler).

### Publishing a release

We use [release-it](https://github.com/webpro/release-it) to automate our release. If you have publish access to the NPM package, run the following from the master branch to publish a new release:

```sh

yarn release

```

NOTE: You must have a `GITHUB\_TOKEN` environment variable available. You can create a GitHub access token with the "repo" access [here](https://github.com/settings/tokens).

## Reporting New Issues

The best way to get your bug fixed is to provide a reduced test case. Please provide a public repository with a runnable example.

## How to Get in Touch

\* Callstack Open Source Discord - [#linaria](https://discord.gg/zwR2Cdh).

## Code Conventions

We use Prettier with ESLint integration for linting and formatting. To fix any auto-fixable lint or matting errors, run:

```sh

yarn lint --fix

```

## License

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