First of all, thank you for contributing. It’s appreciated.

# Design goals

I invite you first to read these design goals. If a PR is rejected, it's on the basis of these goals. Having a PR rejected is a horrible feeling, so before doing work in vain I want to help you understand what defines the xstream codebase.

xstream's goal is to provide a \*\*beginner-friendly foundation for reactive programming\*\*.

- \*\*Beginner-friendly\*\* means:

- It has core operators and extra operators so that the core operators are seen as fundamental building blocks while extra operators exist to support less-common use cases

- It must have few operators (either core or extra) to reduce decision paralysis

- Naming is important to hint what is common and what is rare or wrong. Shorter names mean more common usage, and longer names mean more rare usage. For instance `flatten` is more common to use than `flattenSequentially`, so the former has a shorter name than the latter. Also `shamefullySendNext` is a "bad" use case so it has a bad name too.

- Hot behavior only since it matches people's intuition of a "stream" better

- RefCount with sync start and async stop to support both "shared execution" and "laziness" properties

- \*\*Foundation\*\* means:

- It must be small, its size must float around 4kB minified+gzipped

- It must be fast (enough), competitive with the state of the art reactive programming for JS

- \*\*Reactive programming\*\* means:

- Stream-based programming like in RxJS and most.js

To achieve those above, xstream \*\*sacrifices codebase readability\*\*. We apply hard-coded minification techniques to keep the size as small as possible. A PR to improve readability will be \*\*rejected\*\* if the overall size increases. A PR to improve readability will be \*\*rejected\*\* if it makes the performance slower.

Other reasons why a PR could be rejected:

- Adding a core operator. \*\*Please\*\* open an issue to discuss this before doing any work.

- Rename or change the API.

- Adding user-friendly runtime warnings. Because these would affect code size. We rely on TypeScript to provide type mismatch errors. We are considering adding a development flag for runtime warnings, but we will not accept runtime warnings in production, as this would hurt xstream's "foundation" property.

Note that all of these instructions above are to help you in not doing work in vain. We don't want you to get frustrated trying to contribute to open source.

# To submit a pull request

1. Open a GitHub issue before doing significant amount of work.

2. Clone the repo. If it was already cloned, then git pull to get the latest from master.

4. Run `npm install` before anything else, and wait.

5. Write code.

6. Run `npm test` to lint and test. Don’t commit before fixing all errors and warnings.

7. \*\*Commit using `npm run commit` and follow the CLI instructions.\*\* (important!)

8. Make a pull request.

# To release new versions

You only need to do this if you are an author/publisher of this package.

1. Check that you have npm publishing rights before anything else.

2. Run `npm run check-release`.

3. Run `npm run release`.