# Contributing to FAST

## Getting started

### Machine setup

To work with the FAST [monorepo](https://en.wikipedia.org/wiki/Monorepo) you'll need Git, Node.js, Yarn, and Lerna setup on your machine.

FAST uses Git as its source control system. If you haven't already installed it, you can download it [here](https://git-scm.com/downloads) or if you prefer a GUI-based approach, try [GitHub Desktop](https://desktop.github.com/).

Once Git is installed, you'll also need Node.js, which FAST uses as its JavaScript runtime, enabling its build and test scripts. Node.js instructions and downloads for your preferred OS can be found [here](https://nodejs.org/en/).

Because the FAST repository is structured as a monorepo, we'll need a couple of tools to manage that. The first is Yarn, which can be installed by executing the following command at the terminal:

```shell

npm install -g yarn

```

The second tool you'll need is Lerna, which can be installed with this command:

```bash

yarn global add lerna

```

:::important

The above steps are a one-time setup for your machine and do not need to be repeated after the initial configuration.

:::

### Cloning the repository

Now that your machine is setup, you can clone the FAST repository. Open a terminal and run this command:

```shell

git clone https://github.com/microsoft/fast.git

```

### Installing and building

From within the `fast` folder where you've cloned the repo, install all package dependencies and build all workspaces (local dependencies) with this command:

```bash

yarn

```

After the initial install, you can re-build all workspaces in the future with:

```bash

lerna run prepare

```

### Testing

To run all tests for all packages, use the following command:

```bash

lerna run test

```

This command can also be run from within individual package folders to execute only tests from that package.

:::note

Packages are located within the `packages` folder of the repository. Each package has a `package.json` file with a `scripts` section that defines the commands available to you for common tasks such as build, test, lint, etc.

:::

### Submitting a pull request

If you'd like to contribute by fixing a bug, implementing a feature, or even correcting typos in our documentation, you'll want to submit a pull request. Before submitting a pull request, be sure to [rebase](https://www.atlassian.com/git/tutorials/merging-vs-rebasing) your branch from master. Do not use ``git merge`` or the \*merge\* button provided by GitHub.

### Merging a pull request

If you are merging a pull request, be sure to use the pull request title as the commit title. The title should follow the [conventional commit guidelines](https://www.conventionalcommits.org/). It is recommended that if you are merging in pull requests regularly that you add a browser extension that will auto-correct the title for you. A few that should do this are [Refined GitHub](https://github.com/sindresorhus/refined-github) and [Squashed Merge Message](https://github.com/zachwhaley/squashed-merge-message).

### Documenting breaking changes

Make sure to document the migration strategy in a `MIGRATION.md` file in the package(s) that has breaking changes, eg. `packages/fast-components-styles-msft/MIGRATION.md`.

Example of how to format `MIGRATION.md`:

```md

# Migrating from previous versions

## v1 to v2

- Export `Foo` has been renamed to `Bar`.

- `Bat` has been updated to use the new API [`BatConfig`](link/to/api).

```

## Contribution policy

A “Contribution” is work voluntarily submitted to a project. This submitted work can include code, documentation, design, answering questions, or submitting and triaging issues.

Many contributions require you to agree to a Contributor License Agreement (CLA) declaring that you have the right to grant and do grant the rights to use your contribution. For details, visit [https://cla.microsoft.com](https://cla.microsoft.com).

When you submit a pull request, a CLA-bot automatically determines if you need to provide a CLA and decorates the pull request appropriately (e.g., label, comment). Follow the instructions provided by the bot. You only need to do this once across all repositories using our CLA.

## Guiding principle

Owners, the steering committee, collaborators, code owners, and contributors work in concert with one another on behalf of the FAST community and prioritize the communities interests over their own.

The development, release, and work management processes must reflect this principle. Accepting contributions to the project requires a review by collaborators.

## Governance

### Owners

\*Owners\* have admin access and are responsible for the management, maintenance, and operations of the FAST repository.

### Steering committee

\*Steering committee\* members are key \*collaborators\* who have demonstrated design or technical expertise critical to the driving the FAST project and community forward.

### Collaborators

\*Collaborators\* have write access and have an active and sustained impact on the project and participate in triaging issues, reviewing code, mentoring, and working to improve the architectural quality.

### Code owners

As subject matter experts, \*code owners\* approve pull requests on the packages they own. There is a required minimum of one code owner for each package. \*Code owners\* are listed in [CODEOWNERS](https://github.com/microsoft/fast/blob/master/.github/CODEOWNERS).

### Contributors

\*Contributors\* have read access and can be anyone who has contributed a completed pull request to the project.

### Nominations & appointments

\* To become a \*contributor\*, a community member must have a pull request approved and merged into the FAST project master branch.

\* To become a \*collaborator\*, a \*contributor\* will petition the \*steering committee\* who will approve or deny the request.

\* To become a \*code owner\*, a \*collaborator\* will be (a) nominated by a \*steering committee\* member or (b) petition the \*steering committee\* who will approve or deny the request.

\* To join the \*steering committee\*, a \*collaborator\* will be nominated by a \*steering committee\* member and the \*steering committee\* who will approve or deny the request.

## Acceptance and consensus seeking process

Acceptance of contributions follows the consensus-seeking process.

All pull requests must be approved by a \*collaborator\* before the pull request can be accepted.

Before a pull request is accepted, time should be given to receive input from \*collaborators\* or \*code owners\* with the expertise to evaluate the changes. The amount of time can vary but at least 3 days during the typical working week and 5 days over weekends should be given to account for international time differences and work schedules.

When a pull request : (a) has a significant impact on the project, (b) is inherently controversial, or (c) has not reached consensus with \*collaborators\*; add a "controversial" label to the pull request for the \*steering committee\* to review the pull request. Pull requests labeled with "controversial" are not approved until the \*steering committee\* reviews the issue and makes a decision.

Additionally, \*owners\*, can temporarily enable [interaction limits](https://help.github.com/articles/limiting-interactions-with-your-repository/) to allow a "cool-down" period when hot topics become disruptive.

Specific \*collaborators\* or \*code owners\* can be added to a pull request by including their user alias.

## Stability policy

An essential consideration in every pull request is its impact on the system. To manage impacts, we work collectively to ensure that we do not introduce unnecessary breaking changes, performance or functional regressions, or negative impacts on usability for users or supported partners.

## Developer's Certificate of Origin 1.1

By making a contribution to this project, I certify that:

\* a. The contribution was created in whole or in part by me and I have the right to submit it under the open source license indicated in the file; or

\* b. The contribution is based upon previous work that, to the best of my knowledge, is covered under an appropriate open source license and I have the right under that license to submit that work with modifications, whether created in whole or in part by me, under the same open source license (unless I am permitted to submit under a different license), as indicated in the file; or

\* c. The contribution was provided directly to me by some other person who certified (a), (b) or (c) and I have not modified it.

\* d. I understand and agree that this project and the contribution are public and that a record of the contribution (including all personal information I submit with it, including my sign-off) is maintained indefinitely and may be redistributed consistent with this project or the open source license(s) involved.

## Resources

Several open source projects have influenced our contribution policy:

\* [Project Governance @Node](https://nodejs.org/en/about/governance/)

\* [Contributions @Node](https://github.com/nodejs/node/blob/master/CONTRIBUTING.md)

\* [Open Source @GitHub](https://github.com/blog/2039-adopting-the-open-code-of-conduct)

\* [Open Source examples @todogroup](https://github.com/todogroup/policies)