# Contributing to OpenLayers

Thanks for your interest in contributing to OpenLayers. Please see the project's [Code of Conduct](https://github.com/openlayers/openlayers/blob/main/CODE\_OF\_CONDUCT.md) before proceeding.

## Asking Questions

Please ask questions about using the library on [Stack Overflow using the tag 'openlayers'](http://stackoverflow.com/questions/tagged/openlayers).

When you want to get involved and discuss new features or changes, please use [the mailing list](https://groups.google.com/forum/#!forum/openlayers-dev).

## Submitting Bug Reports

Please use the [GitHub issue tracker](https://github.com/openlayers/openlayers/issues). Before creating a new issue, do a quick search to see if the problem has been reported already.

## Getting Familiar with the Code in the Repository

Look for `readme.md` files! Several directories contain a `readme.md` file that explains the contents of the directory and how to work with them.

## Contributing Code

See [`DEVELOPING.md`](https://github.com/openlayers/openlayers/blob/main/DEVELOPING.md) to learn how to get started developing.

Our preferred means of receiving contributions is through [pull requests](https://help.github.com/articles/using-pull-requests). Make sure

that your pull request follows our pull request guidelines below before submitting it.

This page describes what you need to know to contribute code to OpenLayers as a developer.

## Contributor License Agreement

Your contribution will be under our [license](https://raw.githubusercontent.com/openlayers/openlayers/main/LICENSE.md) as per [GitHub's terms of service](https://help.github.com/articles/github-terms-of-service/#6-contributions-under-repository-license).

## Pull request guidelines

Before working on a pull request, create an issue explaining what you want to contribute. This ensures that your pull request won't go unnoticed, and that you are not contributing something that is not suitable for the project. Once a core developer has set the `pull request accepted` label on the issue, you can submit a pull request. The pull request description should reference the original issue.

Your pull request must:

\* Follow OpenLayers's coding style.

\* Pass the integration tests run automatically by the Continuous Integration system.

\* Address a single issue or add a single item of functionality.

\* Contain a clean history of small, incremental, logically separate commits,

with no merge commits.

\* Use clear commit messages.

\* Be possible to merge automatically.

### Address a single issue or add a single item of functionality

Please submit separate pull requests for separate issues. This allows each to

be reviewed on its own merits.

### Contain a clean history of small, incremental, logically separate commits, with no merge commits

The commit history explains to the reviewer the series of modifications to the

code that you have made and breaks the overall contribution into a series of

easily-understandable chunks. Any individual commit should not add more than

one new class or one new function. Do not submit commits that change thousands

of lines or that contain more than one distinct logical change. Trivial

commits, e.g. to fix lint errors, should be merged into the commit that

introduced the error. See the [Atomic Commit Convention on Wikipedia](http://en.wikipedia.org/wiki/Atomic\_commit#Atomic\_Commit\_Convention) for more detail.

`git apply --patch` and `git rebase` can help you create a clean commit

history.

[Reviewboard.org](http://www.reviewboard.org/docs/codebase/dev/git/clean-commits/)

and [Pro GIT](http://git-scm.com/book/en/Git-Tools-Rewriting-History) have

explain how to use them.

### Use clear commit messages

Commit messages should be short, begin with a verb in the imperative, and

contain no trailing punctuation. We follow

http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html

for the formatting of commit messages.

Git commit message should look like:

Header line: explaining the commit in one line

Body of commit message is a few lines of text, explaining things

in more detail, possibly giving some background about the issue

being fixed, etc etc.

The body of the commit message can be several paragraphs, and

please do proper word-wrap and keep columns shorter than about

74 characters or so. That way "git log" will show things

nicely even when it's indented.

Further paragraphs come after blank lines.

Please keep the header line short, no more than 50 characters.

### Be possible to merge automatically

Occasionally other changes to `main` might mean that your pull request cannot

be merged automatically. In this case you may need to rebase your branch on a

more recent `main`, resolve any conflicts, and `git push --force` to update

your branch so that it can be merged automatically.