# How to Contribute

Thank you for your interest in contributing to PYODIDE! There are many ways to contribute, and we appreciate all of them. Here are some guidelines & pointers for diving into it.

## Development Workflow

See [building from sources](./building\_from\_sources.html) and [testing](./testing.html) documentation.

For code-style the use of [pre-commit](https://pre-commit.com/) is also recommended,

```

pip install pre-commit

pre-commit install

```

This will run a set of linters at each commit. Currently it runs yaml syntax validation and is removing trailing whitespaces.

## Code of Conduct

PYODIDE has adopted a [Code of Conduct](code-of-conduct.html) that we expect all contributors and core members to adhere to.

## Development

Work on PYODIDE happens on Github. Core members and contributors can make Pull Requests to fix issues and add features, which all go through the same review process. We’ll detail how you can start making PRs below.

We’ll do our best to keep `master` in a non-breaking state, ideally with tests always passing. The unfortunate reality of software development is sometimes things break. As such, `master` cannot be expected to remain reliable at all times. We recommend using the latest stable version of PYODIDE.

PYODIDE follows semantic versioning (http://semver.org/) - major versions for breaking changes (x.0.0), minor versions for new features (0.x.0), and patches for bug fixes (0.0.x).

We keep a file, [doc/changelog.md](./changelog.html), outlining changes to PYODIDE in each release. We like to think of the audience for changelogs as non-developers who primarily run the latest stable. So the change log will primarily outline user-visible changes such as new features and deprecations, and will exclude things that might otherwise be inconsequential to the end user experience, such as infrastructure or refactoring.

## Bugs & Issues

We use [Github Issues](https://github.com/iodide-project/pyodide/issues) for announcing and discussing bugs and features. Use [this link](https://github.com/iodide-project/pyodide/issues/new) to report an bug or issue. We provide a template to give you a guide for how to file optimally. If you have the chance, please search the existing issues before reporting a bug. It's possible that someone else has already reported your error. This doesn't always work, and sometimes it's hard to know what to search for, so consider this extra credit. We won't mind if you accidentally file a duplicate report.

Core contributors are monitoring new issues & comments all the time, and will label & organize issues to align with development priorities.

## How to Contribute

Pull requests are the primary mechanism we use to change PYODIDE. GitHub itself has some [great documentation](https://help.github.com/articles/about-pull-requests/) on using the Pull Request feature. We use the "fork and pull" model [described here](https://help.github.com/articles/about-pull-requests/), where contributors push changes to their personal fork and create pull requests to bring those changes into the source repository.

Please make pull requests against the `master` branch.

If you’re looking for a way to jump in and contribute, our list of [`good first issues`](https://github.com/iodide-project/pyodide/labels/good%20first%20issue) is a great place to start.

If you’d like to fix a currently-filed issue, please take a look at the comment thread on the issue to ensure no one is already working on it. If no one has claimed the issue, make a comment stating you’d like to tackle it in a PR. If someone has claimed the issue but has not worked on it in a few weeks, make a comment asking if you can take over, and we’ll figure it out from there.

We use [py.test](https://pytest.org), driving [Selenium](https://www.seleniumhq.org) as our testing framework. Every PR will automatically run through our tests, and our test framework will alert you on Github if your PR doesn’t pass all of them. If your PR fails a test, try to figure out whether or not you can update your code to make the test pass again, or ask for help. As a policy we will not accept a PR that fails any of our tests, and will likely ask you to add tests if your PR adds new functionality. Writing tests can be scary, but they make open-source contributions easier for everyone to assess. Take a moment and look through how we’ve written our tests, and try to make your tests match. If you are having trouble, we can help you get started on our test-writing journey.

All code submissions should pass `make lint`. Python is checked with the default settings of `flake8`. C and Javascript are checked against the Mozilla style in `clang-format`.

## Migrating patches

It often happens that patches need to be migrated between different versions of upstream packages.

If patches fail to apply automatically, one solution can be to

1. Checkout the initial version of the upstream package in a separate repo, and create a branch from it.

2. Add existing patches with `git apply <path.path>`

3. Checkout the new version of the upstream package and create a branch from it.

4. Cherry-pick patches to the new version,

```

git cherry-pick <commit-hash>

```

and resolve conflicts.

5. Re-export last `N` commits as patches e.g.

```

git format-patch -<N> -N --no-stat HEAD -o <out\_dir>

```

## License

All contributions to PYODIDE will be licensed under the [Mozilla Public License 2.0 (MPL 2.0)](https://www.mozilla.org/en-US/MPL/2.0/). This is considered a "weak copyleft" license. Check out the [tl;drLegal entry][] for more information, as well as Mozilla's [MPL 2.0 FAQ](https://www.mozilla.org/en-US/MPL/2.0/FAQ/) if you need further clarification on what is and isn't permitted.

## Get in Touch

- \_\_Gitter:\_\_ Pyodide currently shares the [#iodide](https://gitter.im/iodide-project/iodide) channel over at gitter.im

[tl;drLegal entry]:https://tldrlegal.com/license/mozilla-public-license-2.0-(mpl-2)