# Contributing

- Read the [style guide](./contributing/style\_guide.md).

- Please don't make [the benchmarks](https://deno.land/benchmarks) worse.

- Ask for help in the [community chat room](https://discord.gg/deno).

- If you are going to work on an issue, mention so in the issue comments

\_before\_ you start working on the issue.

- Please be professional in the forums. We follow

[Rust's code of conduct](https://www.rust-lang.org/policies/code-of-conduct)

(CoC) Have a problem? Email ry@tinyclouds.org.

## Development

Instructions on how to build from source can be found

[here](./contributing/building\_from\_source.md).

## Submitting a Pull Request

Before submitting, please make sure the following is done:

1. That there is a related issue and it is referenced in the PR text.

2. There are tests that cover the changes.

3. Ensure `cargo test` passes.

4. Format your code with `./tools/format.py`

5. Make sure `./tools/lint.py` passes.

## Changes to `third\_party`

[`deno\_third\_party`](https://github.com/denoland/deno\_third\_party) contains most

of the external code that Deno depends on, so that we know exactly what we are

executing at any given time. It is carefully maintained with a mixture of manual

labor and private scripts. It's likely you will need help from @ry or

@piscisaureus to make changes.

## Adding Ops (aka bindings)

We are very concerned about making mistakes when adding new APIs. When adding an

Op to Deno, the counterpart interfaces on other platforms should be researched.

Please list how this functionality is done in Go, Node, Rust, and Python.

As an example, see how `Deno.rename()` was proposed and added in

[PR #671](https://github.com/denoland/deno/pull/671).

## Releases

Summary of the changes from previous releases can be found

[here](https://github.com/denoland/deno/releases).

## Documenting APIs

It is important to document public APIs and we want to do that inline with the

code. This helps ensure that code and documentation are tightly coupled

together.

### Utilize JSDoc

All publicly exposed APIs and types, both via the `deno` module as well as the

global/`window` namespace should have JSDoc documentation. This documentation is

parsed and available to the TypeScript compiler, and therefore easy to provide

further downstream. JSDoc blocks come just prior to the statement they apply to

and are denoted by a leading `/\*\*` before terminating with a `\*/`. For example:

```ts

/\*\* A simple JSDoc comment \*/

export const FOO = "foo";

```

Find more at https://jsdoc.app/