# Contributing

Thanks for adding into Navalia and making the world a better place™. Navalia is written in TypeScript, so it's probably best to get somewhat familiar with that language and how it works. You'll also need `NodeJS` installed on your system to be productive.

## TD;DR

- Favor simplicity over verbosity. Many different users will consume this.

- Favor speed over accuracy. Let users opt-in to accuracy over speed.

- Abstract the "inner-workings" away from users. Most people don't care about the mechanics of web browsers.

- Be nice. People have endured enough to tolerate contempt from others.

## Workflow

In order to contribute back into Navalia, you'll have to fork the repository to your own account. Once that is complete you can purse whatever git-style workflow you'd like. We try to prefix branch names with meaningful descriptions like `feature` or `bugfix`, follow by a `/` and the thing you're trying to accomplish.

Here's a few examples: `feature/fast-start`, `bugfix/chrome-wont-start`, and `refactor/chrome-uti`. This helps us get some context on what you're trying to achieve from a high-level.

## Philosophies

Navalia was created to make driving headless browser \_much simpler\_ than it is at present. Whilst other drivers require complex setups and are slow in comparison, Navalia optimizes for the \_common\_ use case over the "power-user".

As an example, lets say we're to create a new driver for a browser. Starting the browser is an asyncronous process, and is apparent to most consumers, however Navalia tries to mask over that. So favor interactions like this:

```js

const { Chrome } = require('chrome');

const chrome = new Chrome();

chrome.goto();

```

Over interactions like this:

```js

const { Chrome } = require('chrome');

const chromeStarter = new Chrome();

chromeStarter.start().then((readyChrome) => readyChrome.connect()).then((chrome) => {

chrome.goto();

});

```

Even though it's more clear what's going on in the bottom example, most users \_don't care\_ about the inner-working of starting a browser, and just want something simple and readable. This does push some complexity to the code, but it's well worth it.

## API Design

If you've got a wild idea about a new API, make sure that the method is clearly-written and easy to understand. Try to use simple english that, while it might not be overly-descriptive, is easy to understand when it's being used:

```js

// Good

chrome.goto(...);

// Bad

chrome.loadPage(...);

```

API's should also optimize for the common-case over the edge-case. At most only one argument should be required, and the rest of the options should be captured in an optional second-argument with more details:

```js

// Good

chrome.goto('www.google.com', { pageload: true, stopOnError: true });

// Bad

chrome.goto({ url: 'www.google.com', pageOpts: { ... }, whenReady: () => {} });

```

Large interfaces make it hard for new users to understand what's going one, whereas simpler ones can foster an understanding of what's happeneing (at the cost of some clarity).

Use careful judgement when your API handles the default case. For example: most of the time users won't need to capture memory snapshots ever 10ms, so if you're writing a feature that does that make sure it doesn't sacrifice performance for accurateness. Let the consumer "opt-in" to accuracy, but default to speed:

```js

// Good

// opt-ing into heap capturing, which is slower

chrome.goto('www.google.com', { heapCapture: true });

// Bad

// forcing users to "turn-off" for a speed bump

chrome.goto('www.google.com', { heapCapture: false });

```

If you have questions or proposals feel free to issue a PR to gather feedback and thoughts (no question is below us). You can also start an issue and we'd be happy to chat about your feature and guide you on crafting it!

## Hints and Tooling

\*\*Running Local Changes\*\*

This package exports an `npm start` command that lets you run arbitrary scripts at any point. For instance, if you have a `test.ts` file, you can simply do:

```bash

$ npm start -- test.ts

```

This is effectively sugar over using the TypeScript `ts` global to execute scripts.

\*\*Tests\*\*

At the time of this writing the majority of the API has no automated test suite. Once we reach a point of higher test-coverage contributors will need to submit tests with any new feature. Tests are written in `Jest`, and can detailed docs on that test-runner can be found [here](https://facebook.github.io/jest/docs/api.html).

\*\*Prettier\*\*

Navalia will run `prettier` prior to commits being added (when `git add`ing files). This is done to achieve consistent formatting and ensure the repo maintains a level of quality in code preservation.

## Be Nice

It's quite sad that this still has to be cited for open-source projects, but be nice! You might be frustrated by lack of support or features, but it's possible that contributor/maintainer lost a loved-one or something else terrible. It's impossible to see the invisible burdens people carry, so let's give each other the benefit of the doubt and allow room for us humans (and cats).