# Contributing Guidelines

These are the contributing guidelines as well as some documentation on how the code is structured. Read up before contributing to make everything as smooth as possible.

## Posting issues

Just common sense; do a quick search before posting, someone might already have created an issue (or resolved the problem!). If you're posting a bug; try to include as much relevant information as possible (ungit version, node and npm version, OS, any Git errors displayed, the output from CLI console and output from the browser console).

## Pull requests

All PRs are automatically published to NPM once merged (see [#823](https://github.com/FredrikNoren/ungit/issues/823)).

There are two things you have to do for all PRs:

- Make sure to include a note in CHANGELOG.md about the change as part of the PR.

- If it's a code change: Bump the version in `package.json` and `package-lock.json`.

- Does the change fundamentally change how people use Ungit: Bump the major version.

- Does the change introduce new features: Bump the minor version.

- Otherwise (bug fixes, tweaks, and refactoring): Bump patch version.

- If the change doesn't affect the product (e.g. you change the README): No need to bump the version.

## Writing plugins

See [PLUGINS.md](PLUGINS.md)

## Developing for Ungit proper

I do accept pull requests, but I also reserve the right to not do so for things I don't think fit with Ungit. If you are developing anything new you should almost always also provide tests for it, preferably click tests but it doesn't hurt to write REST-interface tests as well if applicable. Try to post pull requests early, even at a concept stage, to get feedback and increase chances it's merged.

### What you need to get started

You'll need the same as for running Ungit; node, npm, and git.

### Getting started

To get started developing on Ungit:

1. Make sure you have [node.js](https://nodejs.org/), [npm](https://www.npmjs.com/) and [git](https://git-scm.com/) installed.

2. Clone the repository to a local directory.

3. Run `npm install` to install dependencies.

4. Run `npm run build` to build (compile templates, CSS and JS).

5. Type `npm start` to start ungit, or `npm test` to run tests.

6. (Optional). Run `npm run watch` to automatically rebuild stuff when you change files.

7. Run `npm run lint` to verify your changes conform to the formatting.

8. (Optional). Run `npm run format` to automatically fix formatting and (some) linting issues.

### Run ungit as a standalone application

To provide easier access to launch ungit a standalone application container using [electron](https://electronjs.org/) is available.

#### To get started

1. Follow steps in 'Getting started' to get a development environment ready.

2. Run `npm run electronpackage`. This will create a standalone application package under `build/`

#### Known limitations

1. The current standalone application does not allow you to execute more than one instance.

2. There is no installer package neither automatic update mechanism for standalone application in place yet.

#### Additional notes

1. To create windows package with proper application description on a non-windows platform, [wine](https://www.winehq.org/) is required to be installed. If not, the windows package will be created with default resources.

### Architecture overview

Ungit has two major parts; the server and the UI. The server exposes a REST interfaces, which enables it to be run on a remote server. The UI is a single-page web-app, built using Knockout.js.

### Folders

- `assets/` Raw assets used for development.

- `bin/` "Binary" files, the ungit launcher script and the credentials-helper, which is invoked by git to acquire credentials when using http authentication.

- `clicktests/` [puppeteer](https://pptr.dev/) click test; basically tests that run on the rendered DOM. Since these run all the way, from the DOM down to the server, they're also the most powerful of the tests.

- `components/` This directory contains all view components for Ungit, each of them exposed as an Ungit plugin.

- `public/` The UI web-app.

- `public/css/` CSS generated by the npm build script.

- `public/fonts/` and `public/images/` Assets which are served directly.

- `public/js/` An ungit.js file generated by the npm build script, as well as raven files which handle exception logging.

- `public/less/` Less files, which are the "source" used to generate the CSS.

- `public/source/` JavaScript source code, which is turned into the `public/js/ungit.js` file by the npm build script.

- `public/vendor/` Various 3rd-party libs.

- `source/` Server and shared (i.e. used by both server and UI) source code.

- `test/` Unit tests and REST interface tests.

### Running tests

`npm test` will run both unit tests, REST-interface tests, and click tests. `npm run unittest` only runs the tests in the `test/` folder, `npm run clicktest` runs only the tests in the `clicktests/` folder. Install Mocha (`npm install -g mocha`) to run specific tests in the test folder and get better stack traces: `mocha test/spec.git-api.js`.

### Things to consider when developing

- Try to make everything as touch friendly as possible, for instance, no mouse over tooltips (try to make it clear without that). Everything doesn't adhere 100% to that right now but I'm trying to move it more in that direction.

- Write tests. The most important tests to write are usually the click tests since they will cover the most code (both UI and backend).