- [Writing Code](#writing-code)

- [Setting Up](#setting-up)

- [Committing](#committing)

- [Don't Mutate `options`](#dont-mutate-options)

- [Building](#building)

- [Adding new commands](#adding-new-commands)

- [Testing](#testing)

- [Optional Plugins to install for your code editor for a better developer experience](#optional-plugins-to-install-for-your-code-editor-for-a-better-developer-experience)

- [Prettier](#prettier)

- [TSlint (linter)](#tslint-linter)

## Writing Code

### Setting Up

1. Remove the installed version from NPM: `[sudo] npm rm -g gh`

2. Fork the project and clone it locally: `git clone git@github.com:<your-username>/gh.git`

3. Go to the package folder and create a symlink: `[sudo] npm link`

4. Then you can run commands normally `gh ...`

### Committing

> Following a commit format allows us to automatically publish new builds via continuous integration

- Practically commits will end up looking like this:

```

fix(pull-request): resolves bug where pull request doesn't close

This happened because we weren't passing the right data to Octokit

fix #123

```

- If you are not familiar with this pattern, simply run `npm run commit` which will take you through a helpful interactive semantic commit process

- If you want more info on the commit process, we follow [Angular's Commit Convention](https://github.com/conventional-changelog/conventional-changelog/tree/master/packages/conventional-changelog-angular#readme)

### Don't Mutate `options`

- We use a variable called `options` throughought our code base which holds core variables like flags that we reuse

- We also freeze it for [immutability benefits](https://redux.js.org/faq/immutable-data#what-are-the-benefits-of-immutability) so if you try to mutate it somewhere, it will yell at you

- If you need to modify it please use the [immer](https://immerjs.github.io/immer/docs/introduction) pattern:

```javascript

import { produce } from 'immer'

options = produce(options, draft => {

draft.list = true

})

```

### Building

Since this is a \*\*TypeScript\*\* project, we have to compile the code before running it.

First, if you haven't already done so, install dependencies and create a sym link:

- `npm install`

- `[sudo] npm link`

To run \*\*incrementally\*\* in watch mode:

- `npm run dev`

To run \*\*once\*\*:

- `npm run build`

### Adding new commands

- Copy and edit the content of [Hello World](https://github.com/node-gh/gh/blob/master/src/cmds/hello.ts) example

- Add instructions in [README](https://github.com/node-gh/gh/blob/master/README.md)

## Testing

Please verify that your tests pass & minimum coverage levels are met when contributing code:

- `npm test` Run all tests

- `npm run test:watch` Run all tests in watch mode

- `npm run test:coverage` Run all tests with coverage

- `npm run test pull-request` Run one test

### Optional Plugins to install for your code editor for a better developer experience

- Even if you don't install these plugins, your commits will automatically be formatted

#### Prettier

> Prettier is an opinionated code formatter

- [User Guide](https://prettier.io/)

- [Editor plugin links](https://prettier.io/docs/en/editors.html)

#### TSlint (linter)

> TSlint is a pluggable linting utility for TypeScript

- [User Guide](https://palantir.github.io/tslint/)

- [Editor plugin links](https://palantir.github.io/tslint/usage/third-party-tools)