# Contributing to Luxon

## General guidelines

Patches are welcome. Luxon is at this point just a baby and it could use lots of help. But before you dive in...Luxon is one of those tightly-scoped libraries where the default answer to "should this library do X?" is likely "no". \*\*So ask first!\*\* It might save you some time and energy.

Here are some vague notes on Luxon's design philosophy:

1. We won't accept patches that can't be internationalized using the JS environment's (e.g. the browser's) native capabilities. This means that most convenient humanization features are out of scope.

1. We try hard to have a clear definition of what Luxon does and doesn't do. With few exceptions, this is not a "do what I mean" library.

1. Luxon shouldn't contain simple conveniences that bloat the library to save callers a couple lines of code. Write those lines in your own code.

1. Most of the complexity of JS module loading compatibility is left to the build. If you have a "this can't be loaded in my bespoke JS module loader" problems, this isn't something you should be solving with changes to the `src` directory. If it's a common use case and is possible to generate with Rollup, it can get its own build command.

1. We prefer documentation clarifications and gotchas to go in the docstrings, not in the guides on the docs page. Obviously, if the guides are wrong, they should be fixed, but we don't want them to turn into troubleshooting pages. On the other hand, making sure the method-level documentation has ample examples and notes is great.

1. You'll need to sign a CLA as part of your first pull request to Luxon.

## Building and testing

Building and testing is done through npm scripts. The tests run in Node and require Node 10+ with full-icu support. This is because some of the features available in Luxon (like internationalization and time zones) need that stuff and we test it all. On any platform, if you have Node 10 installed with full-icu, you're good to go; just run npm scripts like `npm run test`. But you probably don't have that, so read on.

### OSX

Mac is easy:

Open the terminal.

```

brew install node --with-full-icu

npm install

./scripts/test

```

If that's for whatever reason a pain, the Linux instructions should also work, as well as the Docker ones.

### Linux

There are two ways to get full-icu support in Linux: build it with that support, or provide it as a module. We'll cover the latter. Assuming you've installed Node 10:

```

npm install

npm install full-icu

./scripts/test

```

Where `scripts/test` is just `NODE\_ICU\_DATA="$(pwd)/node\_modules/full-icu" npm run test`, which is required for making Node load the full-icu module you just installed. You can run all the other npm scripts (e.g. `npm run docs`) directly; they don't require Intl support.

### Windows

If you have [Bash](https://git-scm.com/downloads) or [WSL](https://docs.microsoft.com/en-us/windows/wsl/install-win10), the Linux instructions seem to work fine.

I would love to add instructions for a non-WSL install of the dev env!

### Docker

In case messing with your Node environment just to run Luxon's tests is too much to ask, we've provided a Docker container. You'll need a functioning Docker environment, but the rest is easy:

```

./docker/npm install

./docker/npm run test

```

## Patch basics

Once you're sure your bugfix or feature makes sense for Luxon, make sure you take these steps:

1. Be sure to add tests and run them with `scripts/test`

1. Be sure you run `npm run lint!` before you commit. Note this will modify your source files to line up with the style guidelines.

1. Make sure you add or ESDoc annotations appropriately. You can run `npm run docs` to generate the HTML for them. They land in the `build/docs` directory. This also builds the markdown files in `/docs` into the guide on the Luxon website.

1. To test Luxon in your browser, run `npm run site` and then open `build/demo/global.html`. You can access Luxon classes in the console like `window.luxon.DateTime`.

1. To test in Node, run `npm run build` and then run something like `var DateTime = require('./build/cjs/luxon').DateTime`.

Luxon uses [Husky](https://github.com/typicode/husky) to run the formatter on your code as a pre-commit hook. You should still run `npm run lint!` yourself to catch other issues, but this hook will help prevent you from failing the build with a trivial formatting error.

## npm script reference

| Command | Function |

| ---------------------------- | --------------------------------------- |

| `npm run build` | Build all the distributable files |

| `npm run build-node` | Build just for Node |

| `npm run test` | Run the test suite, but see notes above |

| `npm run format` | Run the Prettier formatter |

| `npm run lint!` | Run the formatter and the linter |

| `npm run docs` | Build the doc pages |

| `npm run site` | Build the Luxon website |

| `npm run check-doc-coverage` | Check whether there's full doc coverage |