\*\*Working on your first Pull Request?\*\* You can learn how from this \_free\_ series

[How to Contribute to an Open Source Project on GitHub](https://egghead.io/series/how-to-contribute-to-an-open-source-project-on-github)

## Contributing to the docs

Documentation uses Sphinx and reStructuredText. Source inside the

[docs](https://github.com/lingui/js-lingui/tree/master/docs) directory.

1. Go to the `docs` directory

2. Run `pipenv install` to setup Python environemnt (requires Python 3.6).

If you encounter `ValueError('unknown locale: %s' % localename)`,

run `export LC\_ALL=en\_US.UTF-8 && export LANG=en\_US.UTF-8` and try again.

3. Run `pipenv run make livehtml` to build the docs, watch for changes and preview

documentation locally at [http://127.0.0.1:8000](http://127.0.0.1:8000).

4. It's also possible to run `pipenv run make html` for single build. Incremental builds

are much faster than the first one as only changed files are built.

## Contributing the code

This project uses [yarn][yarninstall] package manager. Please follow

[official][yarninstall] docs to install it.

### Setup local environment

1. Clone project

```sh

git clone https://github.com/lingui/js-lingui.git

cd js-lingui

```

2. Install development packages. This project uses

[yarn workspaces](https://yarnpkg.com/lang/en/docs/workspaces/) instead of Lerna,

so running `yarn` installs all development packages and also dependencies for all

workspaces (inside `packages/\*`).

```sh

yarn

```

3. Run tests

```sh

# Watch mode

yarn watch

# Single run

yarn test

```

NOTE: if you are using an IDE to run test make sure to use the right Jest config.

For unit tests use `-c scripts/jest/config.unit.js`. Integration tests use

build packages (created using `yarn release:build`) and config `-c scripts/jest/config.integration.js`.

See [package.json](./package.json) for more info.

If you run tests manually instead of using `yarn watch` or `yarn test` commands and your tests

fail due to missing locale data (typically you'll get wrong number and currency formating)

make sure you have `NODE\_ICU\_DATA` variable set: `NODE\_ICU\_DATA=node\_modules/full-icu`.

### Using development version in your project

After you successfully fix a bug or add a new feature, you most probably want

to test it in your project as soon as possible.

`jsLingui` uses [verdaccio](https://verdaccio.org/), a lightweight local NPM registry, to install

local build of packages in examples. You can do the same in your project:

1. Run `verdaccio` locally in docker (follow [verdaccio guide](https://verdaccio.org/docs/en/what-is-verdaccio.html)

if you don't want to run it in Docker):

```sh

docker run -d -it --rm --name verdaccio -p 4873:4873 verdaccio/verdaccio

```

2. Publish local build of packages to registry. Run release script with `--dev` flag:

```sh

yarn release --dev

```

3. Inside your project, run `yarn install @lingui/package@next` to install pre-release

of packages.

4. After you make some changes, you need to run `yarn release --dev` and then

`yalc upgrade` inside your project.

### Finalize changes

Please make sure that all tests pass and linter doesn't report any error before

submitting a PR (Don't worry though! If you can't figure out the problem, create a PR

anyway and we'll help you).

- `yarn lint` - Linting

- `yarn test` - Quick test suite (sufficient)

- `yarn release:test` - Full test suite (recommended)

`yarn release:test` builds all packages, simulates creating packages for NPM, runs unit

tests and finally runs integration tests using production build.

\*\*Note\*\*: Don't commit `scripts/build/results.json` created by `yarn release:test`.

Now you can create PR and let CI service do their work!

If you need any help, just raise an issue or submit an working draft of PR.

[yarninstall]: https://yarnpkg.com/en/docs/install