# Vuelidate Contributing Guide

Hi! We are really excited that you are interested in contributing to Vuelidate. Before submitting your contribution, please make sure to take a moment and read through the following guidelines:

- [Code of Conduct](https://github.com/vuelidate/vuelidate/blob/master/.github/CODE\_OF\_CONDUCT.md)

- [Issue Reporting Guidelines](#issue-reporting-guidelines)

- [Pull Request Guidelines](#pull-request-guidelines)

- [Development Setup](#development-setup)

- [Project Structure](#project-structure)

## Issue Reporting Guidelines

- Make sure that you are familiar with documentation before submitting an issue. [https://vuelidate.netlify.com/](https://vuelidate.netlify.com/)

- Try to provide an example to your issue. Issues without a working fiddle are generally much harder to solve and usually take much more time to actually do it.

## Pull Request Guidelines

- The `master` branch is just a snapshot of the latest stable release. All development should be done in dedicated branches. \*\*Do not submit PRs against the `master` branch.\*\*

- Checkout a topic branch from the relevant branch, e.g. `dev`, and merge back against that branch.

- Work in the `src` folder of packages and \*\*DO NOT\*\* checkin `dist` in the commits.

- It's OK to have multiple small commits as you work on the PR - GitHub will automatically squash it before merging.

- Make sure `npm test` passes. (see [development setup](#development-setup))

- If adding a new feature:

- Add accompanying test case.

- Provide a convincing reason to add this feature. Ideally, you should open a suggestion issue first and have it approved before working on it.

- If fixing bug:

- If you are resolving a special issue, add `(fix #xxxx[,#xxxx])` (#xxxx is the issue id) in your PR title for a better release log, e.g. `update entities encoding/decoding (fix #3899)`.

- Provide a detailed description of the bug in the PR. Live demo preferred.

- Add appropriate test coverage if applicable.

## Development Setup

You will need [Node.js](http://nodejs.org) \*\*version 8+\*\* and [yarn](https://yarnpkg.com/en/docs/install).

After cloning the repo, run:

``` bash

yarn install

```

### Committing Changes

Commit messages should follow the [commit message convention](./COMMIT\_CONVENTION.md) so that changelogs can be automatically generated. Commit messages will be automatically validated upon commit. If you are not familiar with the commit message convention, you can use `npm run commit` instead of `git commit`, which provides an interactive CLI for generating proper commit messages.

### Commonly used NPM scripts

``` bash

# watch and auto re-build a package

$ yarn dev

# watch and auto re-run unit tests of a package

$ yarn test:dev

# build all packages

$ yarn lerna:build

# unit test for all packages

$ yarn lerna:test:unit

# run the full test suite, including linting/type checking for all packages

$ yarn lerna:test

```

There are some other scripts available in the `scripts` section of the `package.json` file.

The default test script will do the following: lint with ESLint -> unit tests with coverage -> e2e tests. \*\*Please make sure to have this pass successfully before submitting a PR.\*\* Although the same tests will be run against your PR on the CI server, it is better to have it working locally.

## Project Structure

The project is split into sub packages, by using a Monorepo methodology. Each package is situated inside a `packages/{name-of-package}` folder.

Each package has slightly different structure, but they all share:

- \*\*src:\*\* The source for the package

- \*\*dist:\*\* The folder holding built packages

- \*\*test:\*\* Holds the tests for the package. Some may have tests near each file.

## Credits

Thank you to all the people who have already contributed to Vuelidate!