# Contributing

Hi there.

Thanks for your interest in Parsley and your will to contribute. You're welcome!

First thing: make sure you are using the \*\*latest official release\*\*.

## Questions?

Please ask questions on [StackOverflow](http://stackoverflow.com/questions/ask) and be sure to include the `parsley.js` tag. Please \*\*provide an example\*\*, starting for example from [this jsfiddle](http://jsfiddle.net/marcandre/58vnaqur/)

## Issues?

If you believe you have found a bug in `parsley`, please \*\*provide an example\*\*, starting for example from [this jsfiddle](http://jsfiddle.net/marcandre/58vnaqur/).

This makes it possible for you to be sure you have isolated the issue to a minimal case. It also makes it much easier for us to understand your issue. Sometimes the issue is [completely different than what you would expect](https://github.com/guillaumepotier/Parsley.js/issues/711) and only an actual example can lead to a solution.

## Pull requests?

To run tests locally:

npm install # needed only the very first time!

gulp test # runs tests in the console

# or to run them in the browser:

gulp test-browser # starts a local server

open test/runner.html # open in your favorite browser

\*Note:\* Parsley is written in EcmaScript 6.

\*Note:\* There's currently a test that fails on some system, a beer to you if you can fix that, see https://github.com/guillaumepotier/Parsley.js/issues/1095

\*\*More tips general to any open source projects\*\*

Here are a few simple rules you'll have to follow in order to ease code reviews,

discussions and PR merging.

1) You MUST follow Parsley coding standard. It basically follows

[these popular rules](http://sideeffect.kr/popularconvention#javascript) except that

functions are followed by one space on Parsley. Use two spaces as tabs.

2) You MUST write / update tests for each code contribution you do

3) You MUST commit a passing test-suite

4) You SHOULD write documentation

5) You MUST NOT modify `/dist`. Changes are made only to `/src`, `/test`, etc...

Please, write [commit messages that make

sense](http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html),

and [rebase your branch](http://git-scm.com/book/en/Git-Branching-Rebasing)

before submitting your Pull Request.

One may ask you to [squash your

commits](http://gitready.com/advanced/2009/02/10/squashing-commits-with-rebase.html)

too. This is used to "clean" your Pull Request before merging it (we don't want

commits such as `fix tests`, `fix 2`, `fix 3`, etc.). Good rule of thumb: the test-suite must pass at each commit point.

Also, while creating your Pull Request on GitHub, you MUST write a description

which gives the context and/or explains why you are creating it.

Thank you!

Inspired by http://williamdurand.fr/2013/07/04/on-open-sourcing-libraries/