Conventional Commits

One of the most difficult things in the world, after flushing the cache and giving the names to the variables, is to understand what changes from one version to another in your code. The use of semantic versioning is a de facto standard, especially in opensource libraries. But after only 3 months, how can we remember the differences between version `2.0.1` and `2.0.2` of the our project?

The use of `CHANGELOG.md` has become more than necessary, both to remember what has changed and to inform those who use our code of what has actually changed. **But I'm a lazy developer**, I will never have the ability to remember what I did at each commit and to remind me on every release to update a file by hand. After googling for hours I found out that the answer I was looking for was right in front of my nose from the beginning. The git's history contains all the information you need to generate a changelog file automatically!

But, before talking about how to automatically create our changelogs I have to introduce you to the conventional commits

The Conventional Commits specification is a lightweight convention on top of commit messages. It provides an easy set of rules for creating an explicit commit history, which makes it easier to write automated tools on top of. This convention dovetails with SemVer, by describing the features, fixes, and breaking changes made in commit messages.

Commits will have a standard structure that serves to describe exactly what happened in that commit: