Conventional Commits

The Conventional Commits 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 describes the features, fixes, and breaking changes made in commit messages:

1. **fix**: a commit of the type fix patches a bug in your codebase.
2. **feat:** a commit of the *type* `feat` introduces a new feature to the codebase.
3. **BREAKING CHANGE:** a commit that has a footer `BREAKING CHANGE:`, or appends a `!` after the type/scope, introduces a breaking API change. A BREAKING CHANGE can be part of commits of any *type*.
4. *types* other than `fix:` and `feat:` are allowed, for example @commitlint/config-conventional (based on the the Angular convention) recommends `build:`, `chore:`, `ci:`, `docs:`, `style:`, `refactor:`, `perf:`, `test:`, and others.
5. *footers* other than `BREAKING CHANGE: <description>` may be provided and follow a convention similar to git trailer format.

Examples

Commit message with description and breaking change footer

> feat: allow provided config object to extend other configs
>
> BREAKING CHANGE: `extends` key in config file is now used for extending other config files

Commit message with ! to draw attention to breaking change

> refactor!: drop support for Node 6

Commit message with both ! and BREAKING CHANGE footer

> refactor!: drop support for Node 6
>
> BREAKING CHANGE: refactor to use JavaScript features not available in Node 6.

Commit message with no body

> docs: correct spelling of CHANGELOG

Commit message with scope

> feat(lang): add polish language

Commit message with multi-paragraph body and multiple footers

> fix: correct minor typos in code

Check here for the full specification.