Git Commit Message Convention

> *This is adapted from [Angular's commit convention](#).*

TL;DR:

Messages must be matched by the following regex:

```
/^(revert: )?(feat|fix|polish|docs|style|refactor|perf|test|workflow|ci|chore|types)
(\(.+\))?: .{1,50}/
```

Examples

Appears under "Features" header, `compiler` subheader:

```
feat(compiler): add 'comments' option
```

Appears under "Bug Fixes" header, `v-model` subheader, with a link to issue #28:

```
fix(v-model): handle events on blur

close #28
```

Appears under "Performance Improvements" header, and under "Breaking Changes" with the breaking change explanation:

```
perf(core): improve vdom diffing by removing 'foo' option

BREAKING CHANGE: The 'foo' option has been removed.
```

The following commit and commit `667ecc1` do not appear in the changelog if they are under the same release. If not, the revert commit appears under the "Reverts" header.

```
revert: feat(compiler): add 'comments' option

This reverts commit 667ecc1654a317a13331b17617d973392f415f02.
```

Full Message Format

A commit message consists of a **header**, **body** and **footer**. The header has a **type**, **scope** and **subject**:

```
<type>(<scope>): <subject>
<BLANK LINE>
<body>
<BLANK LINE>
<footer>
```

The **header** is mandatory and the **scope** of the header is optional.

Revert

If the commit reverts a previous commit, it should begin with `revert: `, followed by the header of the reverted commit. In the body, it should say: `This reverts commit <hash>.`, where the hash is the SHA of the commit being reverted.

Type

If the prefix is `feat`, `fix` or `perf`, it will appear in the changelog. However, if there is any BREAKING CHANGE, the commit will always appear in the changelog.

Other prefixes are up to your discretion. Suggested prefixes are `docs`, `chore`, `style`, `refactor`, and `test` for non-changelog related tasks.

Scope

The scope could be anything specifying the place of the commit change. For example `core`, `compiler`, `ssr`, `v-model`, `transition` etc...

Subject

The subject contains a succinct description of the change:

- use the imperative, present tense: "change" not "changed" nor "changes"
- don't capitalize the first letter
- no dot (.) at the end

Body

Just as in the **subject**, use the imperative, present tense: "change" not "changed" nor "changes". The body should include the motivation for the change and contrast this with previous behavior.

Footer

The footer should contain any information about **Breaking Changes** and is also the place to reference GitHub issues that this commit **Closes**.

Breaking Changes should start with the word `BREAKING CHANGE:` with a space or two newlines. The rest of the commit message is then used for this.