commit 规范

目前比较知名的规范是 `Angular` 的 `commit message` 规范

规范格式

```
<header>
<BLANK LINE>
<body>
<BLANK LINE>
<footer>
```

每一个commit message都应该包含header，body和footer。其中footer可以省略，但是header和body都不能为空。

Header

Header分为三个部分type, scope, summary

```
<type>(<scope>): <summary>
```

- type：用于说明git commit的类别（在做什么事情）
- Scope：用于说明 commit 影响的范围（对什么做）
- summary：对commit的一个简短的描述，一般不超过50字

```
feat(view): add xxx page
```

Header Type:

用于说明git commit的类别，只允许使用下面的标识。

- feat: 新功能（feature）。
- fix: 修复bug，可以是QA发现的BUG，也可以是研发自己发现的BUG。
- docs: 文档（documentation）。
- style: 格式（不影响代码运行的变动）。

Header Type:

- refactor: 重构（即不是新增功能，也不是修改bug的代码变动）。
- perf: 优化相关，比如提升性能、体验。
- test: 增加测试。
- chore: 构建过程或辅助工具的变动。
- revert: 回滚到上一个版本。

Body

和Header中的summary一样。但是是当前操作的详细描述

例如：
解释提交的动机，为什么需要更改，可以和之前的行为进行比较，来说明改动的影响等等。

更多是说明具体变化

Footer

如果有兼容变化或者弃用情况。
Footer部分以BREAKING CHANGE开头，后面是对变动的描述、以及变动理由和迁移方法

同时可以把相关Github issue，JIRA ticket或者其他文档链接填入其中

```
DEPRECATED: <what is deprecated>

<deprecation description + recommended update path>


Closes #<pr number>
```

完整示例

```
git commit -m "feat(users): add user authentication feature

- 完成用户登录、登出相关处理.
- 增加基于JWT的用户状态校验.

Closes #123"
```