目的

- 统一团队git commit标准
- 方便查看每次提交版本更新内容
- 可读性好，方便做`code review`
- 自动生成`CHANGELOG`

原则

- 提交粒度是一个`feature`或一个`bug fix`
- 使用`git cz`严格按照`commit`规范提交
- 不要每`commit`一次就`push`一次，多积攒几个`commit`后一次性`push`

格式

- 每个`commit message`应该包含一个`header`、一个`body`和一个`footer`
- `header`有一个特殊的格式，包含`type`（类型）、`scope`（作用域）和`subject`（主题）
- `header`是必须的，`body`和`footer`是可选的
- 建议提交的说明部分不要超过70个字符，方便阅读

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

<body>

<footer>

Example:

feat(route, controller, service): add users module

add users entity for mutil method apis
```

Type

`type`提交类型，必须是下列之一：

- **build**：影响构建系统或外部依赖的改变
- **ci**：改变CI配置文件和脚本
- **docs**：只改变文档
- **feat**：一个新功能
- **fix**：一个bug fix
- **perf**：代码更改,提高了性能
- **refactor**：代码重构
- **style**：不影响代码含义，只是代码风格的改变
- **test**：添加缺失的测试或修正现有的测试
- **wip**：移除文件或者代码

scope

`scope`是被影响的目录或文件，这样能方便阅读提交日志。

比如，修改的目录为`tools`，那么scope就是`tools`：

```
feat(tools): add a data utils
```

subject

`subject`是`commit`的简短描述：

- 以动词开头，使用第一人称现在时，比如`change`，而不是`changed`或`changes`
- 第一个字母小写
- 结尾不加英文句点(.)

body

`body`是对本次`commit`的详细描述，可以分成多行：

- 使用第一人称现在时，比如`change`，而不是`changed`或`changes`
- 说明代码变动的原因，以及前后对比

footer

- 不兼容变动：如果当前代码和上一个版本不兼容，则`footer`部分以`BREAKING CHANGE`开头，后面是对变动的描述、理由和迁移方法
- 关闭`issue`：如果当前`commit`针对某个`issue`，那么可以在`footer`部分关闭这个`issue`

Revert

还有一种特殊情况，如果当前`commit`用于撤销以前的`commit`，则必须以`revert`开头，后面跟着被撤销`commit`的`header`

```
revert: feat(pencil): add pencil option

This reverts commit 667ecc1654a317a13331b17617d973392f415f02
```

此处`body`部分格式是固定的，必须写成`This reverts commit <hash>`，其中`hash`是被撤销`commit`的SHA标识符

接入

安装

Commitizen

Commitizen是一个格式化`commit message`的工具

```
npm i commitizen -g
```

在项目根目录输入以下命令：

```
commitizen init cz-conventional-changelog --save --save-exact
```

| 注意：`commitizen`是基于`Node.js`的，如果没有`package.json`文件会报错，需要先执行`npm init`初始化

执行以上命令没有报错则表示成功

```
Attempting to initialize using the npm package cz-conventional-changelog
Document@1.0.0 /Users/jisen/Documents/Document
└─┬ cz-conventional-changelog@2.0.0
  ├── conventional-commit-types@2.1.0
  ├── lodash.map@4.6.0
  ├── longest@1.0.1
  ├─┬ pad-right@0.2.2
  │ └── repeat-string@1.6.1
  ├── right-pad@1.0.1
  └── word-wrap@1.2.1
```

当我们需要`commit`的时候，使用`git cz`替换`git commit`命令，会出现提交类型的选择

```
[redacted]$ git cz
[redacted]
cz-cli@2.9.6, cz-conventional-changelog@2.0.0


Line 1 will be cropped at 100 characters. All other lines will be wrapped after
100 characters.

? Select the type of change that you're committing: (Use arrow keys)
❯ feat:     A new feature
  fix:      A bug fix
  docs:     Documentation only changes
  style:    Changes that do not affect the meaning of the code (white-space, for
matting, missing semi-colons, etc)
  refactor: A code change that neither fixes a bug nor adds a feature
  perf:     A code change that improves performance
  test:     Adding missing tests or correcting existing tests
(Move up and down to reveal more choices)
```

然后根据提示进行选择、输入就可以了

注意：如果是第二次配置，需要用`--force`

```
commitizen init cz-conventional-changelog --save --force
```

自动生成`CHANGELOG`

conventional-changelog-cli是自动生成`CHANGELOG`的工具

安装：

```
npm i conventional-changelog-cli -g
```

在`package.json`中添加以下配置：

```
"scripts": {
  "log": "conventional-changelog -p angular -i CHANGELOG.md -s"
}
```

执行 `npm run log` 自动生成`CHANGELOG.md`文件