

# Git commit 格式

## 目的

- 统一团队Git Commit标准，便于后续代码review、版本发布、自动化生成change log；
- 可以提供更多更有效的历史信息，方便快速预览以及配合cherry-pick快速合并代码；
- 团队其他成员进行类git blame时可以快速明白代码用意；

## Git版本规范

### 分支

- master分支为主分支(保护分支)，不能直接在master上进行修改代码和提交；
- develop分支为测试分支，所有开发完成需要提交测试的功能合并到该分支；
- feature分支为开发分支，大家根据不同需求创建独立的功能分支，开发完成后合并到develop分支；
- fix分支为bug修复分支，需要根据实际情况对已发布的版本进行漏洞修复；

### Tag

采用三段式，v版本.里程碑.序号，如v1.2.1

- 架构升级或架构重大调整，修改第1位
- 新功能上线或者模块大的调整，修改第2位
- bug修复上线，修改第3位

具体操作，可参见：[Git标签](#)、[Git基础-打标签](#)

## changelog

版本正式发布后，需要生产changelog文档，便于后续问题追溯。

## 安装 commitizen

```
npm install -g commitizen
```

## 安装 cz-conventional-changelog

```
npm install -g cz-conventional-changelog
```

## 配置 commitizen adapter

```
echo '{ "path": "cz-conventional-changelog" }' > ~/.czrc
```

# commitizen详解

## Message 格式

一般来说，Commit Message 应包含三部分内容：Header、Body、Footer

```
1 <type>(<scope>): <subject>
2 // 空一行
3 <body>
4 // 空一行
5 <footer>
```

## Header

Header部分应只包含一行，包括三个字段：type、scope和subject

- type type用于说明Commit的类型，包含一下7种类型

```
1 feat: 新功能 (feature)
2 fix: 修补bug
3 docs: 文档 (documentation)
4 style: 格式 (不影响代码运行的变动)
5 refactor: 重构 (即不是新增功能，也不是修改bug的代码变动)
6
7 perf: 优化性能 (a code change that improves performance)
test: 增加测试
chore: 构建过程或辅助工具的变动
```

- scope

```
1 scope用于说明本次Commit所影响的范围，比如controller、user或者README，视项目的
```

- subject

subject是本次Commit目的的简短描述，一般不要超过50个字符

```
1 以动词开头，使用第一人称现在时，比如change，而不是changed或changes
2 第一个字母小写
3 结尾不加句号 (.)
```

## Body

Body是对本地提交的一个详细描述，下面是一个示例

```
1 More detailed explanatory text, if necessary. Wrap it to
2 about 72 characters or so.
3
4 Further paragraphs come after blank lines.
5
6 - Bullet points are okay, too
7 - Use a hanging indent
```

## Footer

Footer只用于两种情况

- 不兼容改动

如果当前代码与上一个版本不兼容，则 Footer 部分以BREAKING CHANGE开头，后面是对变动的描述、以及变动理由和迁移方法。






- 关闭Issue

如果当前Commit是针对某个Issue的提交，那么久可以在Footer中关闭这个Issue：Closes #234



## Git提交信息

message信息格式采用目前主流的**Angular**规范，这是目前使用最广的写法，比较合理和系统化，并且有配套的工具。




Commits on Jan 5, 2016

-  docs(error/\$rootScope/inprog): add missing "\$timeout" ...  
wytesk133 committed with petebacondarwin 10 days ago
-  docs(tutorial/2): add e2e test missing filename ...  
monkpit committed with petebacondarwin 3 hours ago
-  revert: feat(\$compile): Allow ES6 classes as controllers with `bindTo...` ...  
petebacondarwin committed 3 hours ago
-  fix(\$animateCss): only (de)register listeners when events have been a... ...  
Narretz committed 25 days ago
-  fix(loader): use `false` as default value for `transclude` in compone... ...  
sarod committed with petebacondarwin 18 days ago

Commits on Jan 3, 2016

-  feat(\$compile): Allow ES6 classes as controllers with `bindToControll...` ...  
Igalfaso committed 21 days ago
-  chore(\*): Updated year in licence ...  
leuanG committed with Igalfaso 3 days ago

Commits on Jan 1, 2016

-  docs: reorganize information about interpolation ...  
Narretz committed on Dec 6, 2015
-  docs: add docs for ngPattern, ngMinlength, ngMaxlength ...  
Narretz committed on Sep 24, 2015
-  docs(\$interpolateProvider): remove superfluous ng-app attribute ...  
Narretz committed 4 days ago

## 生成change log文件

cz-conventional-changelog 可以自动根据提交信息生成change log，便于统一管理和查阅！

```
1
2 $ npm install -g conventional-changelog-cli
3
```

进入项目执行

```
1 # 在之前生成的基础上，叠加
2 $ conventional-changelog -p angular -i CHANGELOG.md -s
3 # 生成所有记录，包括之前的
4 $ conventional-changelog -p angular -i CHANGELOG.md -s -r 0
```

同样可以创建npm脚本，来更方便的操作

```
1 "scripts": {
2   "changelog": "conventional-changelog -p angular -i CHANGELOG.md -s -r 0 && git add
```

```
CHANGELOG.md"
```

```
3 }
```

强制验证提交信息

采用Git hooks来拦截提交信息，进行格式判断。这里使用commit-msg钩子，该钩子接收一个参数（存有当前提交信息的临时文件的路径）。如果该钩子脚本以非0退出，Git将放弃提交。

yorkie用于执行git-hooks，首先在package.json中增加相关配置

```
1
2 $ npm i --D yorkie
3
```

```
1 "gitHooks": {
2   "commit-msg": "node git-hooks/verify-commit-msg.js"
3 }
```

verify-commit-msg.js

```
1 const chalk = require('chalk')
2 const msgPath = process.env.GIT_PARAMS
3 const msg = require('fs').readFileSync(msgPath, 'utf-8').trim()
4 const versionRE = /^[vV]\d+\.\d+\.\d+/
5 const commitRE = /^(revert: )?
  (feat|fix|docs|style|refactor|perf|test|workflow|ci|chore|types|build|revert)(\(.+\))?: .
  {1,50}/
6
7 if (msg.trimLeft().substring(0, 6) !== 'Merge ' && !commitRE.test(msg) &&
  !versionRE.test(msg)) {
8   console.log()
9   console.log('msg:', msg)
10  console.error(
11    ` ${chalk.bgRed.white(' ERROR ')} ${chalk.red(`invalid commit message format.`)}\n\n` +
12    chalk.red(` Proper commit message format is required for automated changelog
  generation. Examples:\n\n`) +
13    ` ${chalk.green(`feat(compiler): add 'comments' option`)}\n\n` +
14    ` ${chalk.green(`fix(v-model): handle events on blur (close #28)`)}\n\n` +
15    chalk.red(` See .github/COMMIT_CONVENTION.md for more details.\n`) +
16    chalk.red(` You can also use ${chalk.cyan(`npm run commit`)} to interactively generate
  a commit message.\n`)
17  )
18  process.exit(1)
19 }
```