

《Git Commit 规范》

1.引言

来源知乎：

在日常的开发工作中，我们通常使用 git 来管理代码，当我们对代码进行某项改动后，都可以通过 git commit 来对代码进行提交。









git 规定提交时必须写提交信息，作为改动说明，保存在 commit 历史中，方便回溯。规范的 log 不仅有助于他人 review, 还可以有效的输出 CHANGELOG，甚至对于项目的研发质量都有很大的提升。

但是在日常工作中，大多数同学对于 log 信息都是简单写写，没有很好的重视，这对于项目的管理和维护来说，无疑是不友好的。

本篇文章主要是结合我自己的使用经验来和大家分享一下 git commit 的一些规范，让你的 log 不仅“好看”还“实用”。

2.规范 Commit 的好处

看一个不太规范的 commit 记录

 src	init commit	12 months ago
 tests/unit	init commit	12 months ago
 .gitignore	init commit	12 months ago
 README.md	removed readme message	12 months ago
 index.html	init commit	12 months ago
 package.json	changed package.json	12 months ago
 tsconfig.json	init commit	12 months ago
 webpack.config.js	init commit	12 months ago

在看一个规范的commit

<div> <div>dev</div> <div>61 branches</div> <div>250 tags</div> </div> <div> <div>Go to file</div> <div>Add file</div> <div>Code</div> </div>		
<div> <div>vue-bot chore: update sponsors [ci skip] (#11791)</div> <div>4f81b5d on 24 Nov</div> <div>3,131 commits</div> </div>		
.circleci	build: trigger regression tests from 2.6 feature branch	2 years ago
.github	doc(contributing): add missing donation link (#11692)	3 months ago
benchmarks	chore(benchmarks): use flex-wrap (#10380)	17 months ago
dist	build: build 2.6.12	4 months ago
examples	fix(transition): fix appear check for transition wrapper components (#...	2 years ago
flow	chore: spelling and grammar (#11481)	6 months ago
packages	build: build 2.6.12	4 months ago
scripts	chore: remove unused build alias (#9525)	2 years ago
src	build: unexpected token when running dev (#11704)	3 months ago
test	fix(v-pre): do not alter attributes (#10088)	3 months ago
types	fix(types): allow string for watch handlers in options (#10396)	3 months ago
.babelrc.js	workflow: upgraded to babel 7 (#8948)	2 years ago
.editorconfig	chore: https link to editorconfig.org (#11267)	9 months ago
.eslintignore	ignore packages in eslint	5 years ago

总结一下好处：

- 1.规范Commit 提交，可向同事、公众与其他利益关系人传达变化的性质, 对于项目的管理和维护来说比较友好
- 2.利于bug修复的回溯，对于项目的研发质量有比较大的提升
- 3.提供更明确的历史信息，方便判断提交目的和浏览，有助于他人code review, 还可以有效的输出CHANGELOG
- 4.可以过滤某些commit（比如文档改动），便于快速查找信息
- 5.提供了更多的历史信息，方便快速浏览

.....

2.Angular团队Commit Message规范

中文

<类型>[可选的作用域]: <描述>

[可选的正文]

[可选的脚注]

英文

```
<type>[scope]: <subject>
```

```
[body]
```

```
[footer]
```

例如:

添加了一个按钮组件

```
feat(button): add button component
```

```
src/components 中添加了 button 组件
```

```
''
```

修复按钮组件的bug

```
fix(size): fix button size props
```

```
src/components/button 修改了 button 组件的size属性，增加了数字类型的判断
```

```
''
```

type

用于说明 commit 的类型

```
build: 主要目的是修改项目构建系统(例如 glup, webpack, rollup 的配置等)的提交
ci: 主要目的是修改项目继续集成流程(例如 Travis, Jenkins, GitLab CI, Circle等)的提交
docs: 文档更新
feat: 新增功能
merge: 分支合并 Merge branch ? of ?
fix: bug 修复
perf: 性能, 体验优化
refactor: 重构代码(既没有新增功能, 也没有修复 bug)
style: 不影响程序逻辑的代码修改(修改空白字符, 格式缩进, 补全缺失的分号等, 没有改变代码逻辑)
test: 新增测试用例或是更新现有测试
revert: 回滚某个更早之前的提交
chore: 不属于以上类型的其他类型
```

scope

用于说明 commit 影响的范围, 比如: views, component, utils, test...

subject

commit 目的 的简短描述

body

对本次 commit 修改内容的具体描述, 可以分为多行。

body

side effects

....

```
fix(size): fix button size props
```

```
src/components/button 修改了 button 组件的size属性, 增加了数字类型的判断  
可填会不会有其它副作用 影响
```

```
''
```

footer

一些备注, 通常是 `BREAKING CHANGE` (当前代码与上一个版本不兼容) 或修复的 bug(关闭 Issue) 的链接。

break changes

```
fix(size): fix button size props
```

```
src/components/button 修改了 button 组件的size属性, 增加了数字类型的判断
```

```
'可填有没有破坏性的修复 或者 #999'
```

3.项目集成 Git Commit 规范

1.husky

[Husky](#) 是一个增强的 `git hook` 工具。可以在 `git hook` 的各个阶段执行我们在 `package.json` 中配置好的 `npm script`。这些脚本它能阻止不合格的 `git commit`, `git push` 操作。`webpack`、`babel`、`create-react-app`、`antd` 都使用了 `husky`。

Git Hooks, 能帮助我们在代码提交 (`commit`) 和推送时 (`push`) 做一些事情。

Husky 中文文档: <https://www.breword.com/typicode-husky>

需要 `Node >= 10` 和 `Git >= 2.13.0`

1.安装

```
npm install husky@4.3.0 --save-dev
```

2.配置 package.json

```
// package.json
{
  "husky": {
    "hooks": {
      "pre-commit": "npm run lint && npm run test",
      "commit-msg": "",
      "...": "..."
    }
  }
}
```

Git Hooks 的说明:

pre-commit : 预提交消息。该钩子由[git-commit 1]调用, 并且可以通过该 `--no-verify` 选项绕过。它不带任何参数, 并且在获取建议的提交日志消息并进行提交之前被调用。

commit-msg : 提交消息。该钩子由[git-commit 1]和[git-merge 1]调用, 并且可以通过该 `--no-verify` 选项绕过。该挂钩允许在适当位置编辑消息文件, 并可用于将消息规范化为某些项目标准格式。在检查消息文件之后, 它也可以用于拒绝提交。

pre-push : 预推送。这个钩子被[git-push 1]调用, 可以用来防止进行推送。该挂钩由两个参数调用, 这些参数提供目标远程服务器的名称和位置, 如果未使用命名远程服务器, 则两个值将相同。例如: `git push origin master`

Husky 支持的Git Hooks : <https://git-scm.com/docs/githooks>

3.测试

```
PS E:\liujun\workspace\git-commit-demo> git add .
PS E:\liujun\workspace\git-commit-demo> git commit -m"init project"
husky > pre-commit (node v14.15.1)

> vue-pro-temp@0.1.0 lint E:\liujun\workspace\gz-
vue\zggyss3\publiclitigation\gzmx-litigation-vue-web
> vue-cli-service lint
```

执行 git commit 之前会执行: `npm run lint && npm run test` 脚本

2.commitlint

commitlint checks if your commit messages meet the [conventional commit format](#).

commitlint 是一个提交验证工具。原理是可以在实际的 git commit 提交到仓库之前使用 git 钩子来验证信息。提交不符合规则的信息将会被阻止提交到仓库。

对于 `Conventional Commits` 规范, 社区已经整理好了 `@commitlint/config-conventional` 包, 我们只需要安装并启用它就可以了。

commitlint 搭配 husky 的 commit-msg 钩子后, 每次提交 git 版本信息的时候, 会根据配置的规则进行校验, 若不符合规则会 commit 失败, 并提示相应信息。

<https://github.com/conventional-changelog/commitlint>

1.安装 commitlint 依赖

```
npm install @commitlint/config-conventional@^11.0.0 @commitlint/cli@^11.0.0 --save-dev
```

2.新建 commitlint.config.js 文件

[@commitlint/config-conventional](#) 默认的配置

[rules](#) 的详细说明

rules: { "规则名": [规则值, 规则配置] }

规则值:

"off"或者0 //关闭规则 "warn"或者1 //打开规则, 并且作为一个警告, 字体颜色为黄色 (并不会导致检查不通过) "error"或者2 //打开规则, 并且作为一个错误, 字体颜色为红色(退出码为1, 检查不通过)

```
// commitlint.config.js
module.exports = {
  // extends: ['@commitlint/config-conventional'],
  rules: {
    'body-leading-blank': [1, 'always'],
    'footer-leading-blank': [1, 'always'],
    'header-max-length': [2, 'always', 72],
    'scope-case': [2, 'always', 'lower-case'],
    'subject-case': [1, 'never', ['sentence-case', 'start-case', 'pascal-case', 'upper-case']],
    'subject-empty': [2, 'never'],
    'subject-full-stop': [2, 'never', '.'],
    'type-case': [2, 'always', 'lower-case'],
    'type-empty': [2, 'never'],
    'type-enum': [
      2,
      'always',
      ['build', 'chore', 'ci', 'docs', 'feat', 'fix', 'perf', 'refactor', 'revert', 'style', 'test', 'improvement']
    ]
  }
}
```

body-leading-blank:[level,rule,value]

- **condition:** body should have a leading blank line
- **level:** warning or 1
- **rule:** always or never
- **value**

type-enum

build: 主要目的是修改项目构建系统(例如 `glup`, `webpack`, `rollup` 的配置等)的提交
ci: 主要目的是修改项目继续集成流程(例如 `Travis`, `Jenkins`, `GitLab CI`, `Circle`等)的提交
docs: 文档更新
feat: 新增功能
merge: 分支合并 Merge branch ? of ?
fix: bug 修复
perf: 性能, 体验优化
refactor: 重构代码(既没有新增功能, 也没有修复 bug)
style: 不影响程序逻辑的代码修改(修改空白字符, 格式缩进, 补全缺失的分号等, 没有改变代码逻辑)
test: 新增测试用例或是更新现有测试
revert: 回滚某个更早之前的提交
chore: 不属于以上类型的其他类型

3.配置 package.json 文件

```
{
  "husky": {
    "hooks": {
      "pre-commit": "npm run lint && npm run test",
      "commit-msg": "commitlint -E HUSKY_GIT_PARAMS"
    }
  }
}
```

commitlint -E : check message in the file at path given by environment variable value

4.测试

正确:

```
PS E:\liujun\workspace\git-commit-demo> git add .

PS E:\liujun\workspace\git-commit-demo> git commit -m"
>> fix(button): fix button size
>>
>> fix button size bug
>>
>> #999 "
```

husky > pre-commit (node v14.15.1)

```
> PS E:\liujun\workspace\git-commit-demo>
> vue-cli-service lint
DONE No lint errors found!
husky > commit-msg (node v14.15.1)
[web_dev1.0 7784bcb] fix(button): fix button size
7 files changed, 1421 insertions(+), 447 deletions(-)
create mode 100644 gzm-x-litigation-vue-web/commitlint.config.js
```

错误:

```
PS E:\liujun\workspace\git-commit-demo> git commit -m"修改了反馈对话框的bug"
```

```

husky > pre-commit (node v14.15.1)

> vue-pro-temp@0.1.0 lint E:\liujun\workspace\gz-
vue\zggyss3\publiclitigation\gzm-x-litigation-vue-web
> vue-cli-service lint

DONE No lint errors found!
husky > commit-msg (node v14.15.1)
✖ input: 修改了反馈对话框的bug
✖ subject may not be empty [subject-empty]
✖ type may not be empty [type-empty]

✖ found 2 problems, 0 warnings
① Get help: https://github.com/conventional-changelog/commitlint/#what-is-commitlint

husky > commit-msg hook failed (add --no-verify to bypass)

```

3.commitizen

commitizen 是一款可以交互式建立提交信息的工具。它帮助我们 从 type 开始一步步建立提交信息。

commitizen: Angular 团队的提交代码方式，后来被大家广泛的应用。

1.安装commitizen

```
npm install --save-dev commitizen@^4.2.2
```

2.添加脚本

```

"scripts": {
  "serve": "vue-cli-service serve",
  "build": "vue-cli-service build",
  "commit": "cz",
  "lint": "vue-cli-service lint"
},

```

```
"commit": "cz",
```

3.配置 commit 常规的适配器

```

// .czrc.json
{
  "path": "cz-conventional-changelog"
}

// or

// package.json
{
  "config": {

```



```

    "commitizen": {
      "path": "cz-conventional-changelog"
    }
  }
}

```

commitizen 根据不同的 adapter 配置 commit message。例如，要使用 Angular 的 commit message 格式，可以安装 `cz-conventional-changelog`。

4.测试

```

PS E:\liujun\workspace\git-commit-demo> git add .
PS E:\liujun\workspace\git-commit-demo> yarn commit
yarn run v1.15.2
$ cz
cz-cli@4.2.2, cz-conventional-changelog@3.3.0

? 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,
formatting, 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)

```

完整的例子:

```

? 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,
formatting, 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
  build:     Changes that affect the build system or external dependencies
(example scopes: gulp, broccoli, npm)
  ci:        Changes to our CI configuration files and scripts (example scopes:
Travis, Circle, BrowserStack, SauceLabs)
  chore:     Other changes that don't modify src or test files
  revert:    Reverts a previous commit
(Move up and down to reveal more choices)

1. Select the type of change that you're committing: fix:      A bug fix
2. What is the scope of this change (e.g. component or file name): (press enter
to skip) button
3. Write a short, imperative tense description of the change (max 87 chars):

```

```
(15) add button size
4. Provide a longer description of the change: (press enter to skip)
   button 添加了 size 属性
5. Are there any breaking changes? Yes
6. Describe the breaking changes:
   没有
7. Does this change affect any open issues? Yes
8. Add issue references (e.g. "fix #123", "re #123").:
   #999
```

4.生成changelog

如果你的所有 Commit 都符合 Angular 格式，那么发布新版本时，Change log 就可以用脚本自动生成

1.安装

```
npm install conventional-changelog-cli@^2.1.1 --save-dev
```

2.配置package.json

```
"scripts": {
  "serve": "vue-cli-service serve",
  "build": "vue-cli-service build",
  "commit": "cz",
  "log": "conventional-changelog -p angular -i CHANGELOG.md -s",
  "lint": "vue-cli-service lint"
},
```

```
"log": "conventional-changelog -p angular -i CHANGELOG.md -s",
```

3.测试

```
yarn genlog
```

4.编写脚本来执行：genlog (可以修改生成文件的名称)

```
"scripts": {
  "serve": "vue-cli-service serve",
  "build": "vue-cli-service build",
  "commit": "cz",
  "genlog": "node ./script/genlog.js",
  "lint": "vue-cli-service lint"
},
```

5.genlog.js 脚本

```
// script/genlog.js

// 自动生成 git commit 提交记录
const cp = require('child_process')

// exec
// eslint-disable-next-line handle-callback-err
cp.exec(`conventional-changelog -p angular -i CHANGELOG-${getNowTime()}.md -s`,
(err, stdout, stderr) => {
  console.log(stdout)
})

function getNowTime () {
  const date = new Date()
  const year = date.getFullYear()
  const month = date.getMonth() + 1
  const day = date.getDate()
  return year + '-' + month + '-' + day
}
```

6.测试

```
PS E:\liujun\workspace\git-commit-demo> yarn genlog
yarn run v1.15.2
$ node ./script/genlog.js

Done in 1.19s.
```

7.生成的文档如下格式:

[CHANGELOG-2020-12-29.md](#)

0.1.0 (2020-12-29)

Bug Fixes

- **btn:** fix btn type ([0495611](#))
- **btn:** fix btn disabled ([c79ad53](#))
- **button:** fix button size ([7784bcb](#)), closes [#999](#)

参考文章:

<https://juejin.cn/post/6844903871832145927> http://www.ruanyifeng.com/blog/2016/01/commit_message_change_log.html

https://zhuanlan.zhihu.com/p/100574040?utm_source=qq