人人贷大前端技术中心
2019年06月20日 阅读 5260

关注

Git commit message 规范

> git是现在市面上最流行的版本控制工具，书写良好的commit message能大大提高代码维护的效率。但是在日常开发中由于缺少对于commit message的约束，导致填写内容随意、质量参差不齐，可读性低亦难以维护。在项目中引入commit message规范已是迫在眉睫。

用什么规范?

现在市面上比较流行的方案是 `约定式提交规范` （`Conventional Commits`），它受到了 `Angular提交准则` 的启发，并在很大程度上以其为依据。 `约定式提交规范` 是一种基于提交消息的轻量级约定。它提供了一组用于创建清晰的提交历史的简单规则；这使得编写基于规范的自动化工具变得更容易。这个约定与 `SemVer` 相吻合，在提交信息中描述新特性、bug 修复和破坏性变更。它的 message 格式如下:

复制代码

```
<类型>[可选的作用域]: <描述>

[可选的正文]

[可选的脚注]
```

Quick Start

1. 全局安装commitizen & cz-conventional-changelog

`commitizen` 是一个撰写合格 `commit message` 的工具，用于代替 `git commit` 指令，而 `cz-conventional-changelog` 适配器提供conventional-changelog标准（约定式提交标准）。基于不同需求，也可以使用不同适配器。

复制代码

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

安装完毕后，可直接使用 `git cz` 来取代 `git commit` 。

全局模式下，需要 `~/.czrc` 配置文件，为 `commitizen` 指定 `Adapter` 。

2. 项目内安装commitlint & husky

`commitlint` 负责用于对 `commit message` 进行格式校验， `husky` 负责提供更易用的 `git hook` 。

`Use npm`

复制代码

```
npm i -D husky @commitlint/config-conventional @commitlint/cli
```

`Use yarn`

复制代码

```
yarn add husky @commitlint/config-conventional @commitlint/cli -D
```

`commitlint` 只能做格式规范，无法触及内容。对于内容质量的把控只能靠我们自己。

3. 添加相应配置

创建 `commitlint.config.js`

复制代码

```
# In the same path as package.json

echo 'module.exports = {extends: ["@commitlint/config-conventional"]};' > ./commitlint.config.js
```

引入 `husky`

复制代码

```
# package.json

...,
"husky": {
    "hooks": {
      "commit-msg": "commitlint -e $GIT_PARAMS"
    }
}
```

4. 使用

执行 `git cz` 进入interactive模式，根据提示依次填写

复制代码

```
1.Select the type of change that you're committing 选择改动类型（<type>）
```


目录

```
2.What is the scope of this change (e.g. component or file name)? 填写改动范围 (<scope>)

3.Write a short, imperative tense description of the change: 写一个精简的描述 (<subject>)

4.Provide a longer description of the change: (press enter to skip) 对于改动写一段长描述 (<body>)

5.Are there any breaking changes? (y/n) 是破坏性修改吗？默认n (<footer>)

6.Does this change affect any openreve issues? (y/n) 改动修复了哪个问题？默认n (<footer>)
```

生成的commit message格式如下：

复制代码

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

填写完毕后，`husky` 会调用 `commitlint` 对message进行格式校验，默认规定 `type` 及 `subject` 为必填项。

任何 `git commit` 指令的 `option` 都能用在 `git cz` 指令上，例如 `git cz -a`

Commit message规范在rrd-fe落地使用情况

针对团队目前使用的情况，我们讨论后拟定了 `commit message` 每一部分的填写规则。

1. type

`type` 为必填项，用于指定commit的类型，约定了 `feat` 、 `fix` 两个 `主要type` ，以及docs、style、build、refactor、revert五个 `特殊type` ， `其余type` 暂不使用。

复制代码

```
# 主要type
feat:     增加新功能
fix:      修复bug

# 特殊type
docs:     只改动了文档相关的内容
style:    不影响代码含义的改动，例如去掉空格、改变缩进、增删分号
build:    构造工具的或者外部依赖的改动，例如webpack，npm
refactor: 代码重构时使用
revert:   执行git revert打印的message

# 暂不使用type
test:     添加测试或者修改现有测试
perf:     提高性能的改动
ci:       与CI（持续集成服务）有关的改动
chore:    不修改src或者test的其余修改，例如构建过程或辅助工具的变动
```

当一次改动包括 `主要type` 与 `特殊type` 时，统一采用 `主要type` 。

2. scope

`scope` 也为必填项，用于描述改动的范围，格式为项目名/模块名，例如： `node-pc/common` `rrd-h5/activity` ，而 `we-sdk` 不需指定模块名。如果一次commit修改多个模块，建议拆分成多次commit，以便更好追踪和维护。

3. body

`body` 填写详细描述，主要描述 `改动之前的情况` 及 `修改动机` ，对于小的修改不作要求，但是重大需求、更新等必须添加body来作说明。

4. break changes

`break changes` 指明是否产生了破坏性修改，涉及break changes的改动必须指明该项，类似版本升级、接口参数减少、接口删除、迁移等。

5. affect issues

`affect issues` 指明是否影响了某个问题。例如我们使用jira时，我们在 `commit message` 中可以填写其影响的 `JIRA_ID` ，若要开启该功能需要先打通 `jira` 与 `gitlab` 。参考文档：docs.gitlab.com/ee/user/pro...

填写方式例如：

复制代码

```
re #JIRA_ID
fix #JIRA_ID
```

示例

- 完整的commit message示例

- 相应的git log

```
commit c0eb4a6cb96b97924835ef5f562578ba7874f335 (HEAD -> fe_qa)
Author: Jibacat <wfwangchengqian@126.com>
Date:   Thu Jun 20 15:04:12 2019 +0800

    build(package.json): 修改typescript版本到3.4.1

    为了支持"项目引用"等新特性，需要修改typescript版本到3.x

    BREAKING CHANGE: 随着typescript版本的更新，项目中使用的escapeIdentifier方法将不再支持。

    re #P2P-0001
```

最后惯例，欢迎大家star我们的人人贷大前端团队博客，所有的文章还会同步更新到知乎专栏 和 掘金账号，我们每周都会分享几篇高质量的大前端技术文章。如果你喜欢这篇文章，希望能动动小手给个赞。

扩展阅读

conventional commits 必读 介绍约定式提交标准。

Angular规范 必读 介绍Angular标准每个部分该写什么、该怎么写。

@commitlint/config-conventional 必读 介绍commitlint的校验规则config-conventional，以及一些常见passes/fails情况。

> **关注下面的标签，发现更多相似文章**

Git

人人贷大前端技术中心 Lv3 大前端开发工程师 @ 人人贷
发布了 28 篇专栏 · 获得点赞 1,284 · 获得阅读 55,624

关注

> 安装掘金浏览器插件
> 打开新标签页发现好内容，掘金、GitHub、Dribbble、ProductHunt 等站点内容轻松获取。快来安装掘金浏览器插件获取高质量内容吧！

评论

输入评论...

刘小夕 Lv4 会拍照的程序媛 @ 京东
作者您好，请问是否可以在公众号转载本篇文章，会在文首标明作者和文章来源~阅读原文也会加上本文的链接~
1天前 回复

haxifang_aircos Lv1 服务端 @ Hash fang
有帮助到我，thank you
1月前 回复

Fine_ Lv1 前端开发工程师
原来git有这么多黑科技 😊
10月前 回复

hezf Lv2 摸鱼 @ 长春丽明
这个不错，滴滴前端的也是这种，cz
10月前 回复

前端雷布斯 前端工程师
赞~
10月前 回复

Jess Lv1 前端码农
👏👏
10月前 回复

相关推荐

专栏 · 巴格梅克 · 3天前 · Git
【译】防止 Git 泄漏的 5 种最佳做法
3

专栏 · AhuntSun · 4天前 · Git
Git应用详解第九讲：Git cherry-pick与Git rebase
1 3

专栏 · zhangbao90s · 17天前 · Git
[译] 图解常用的 Git 指令含义
21 4

专栏 · coder-pig · 1月前 · Git
《吐血整理》一篇文章教你学废Git版本管理
44 10

专栏 · CRPER · 4月前 · Git / GitHub
高频使用的 Git 命令
558 19

专栏 · Linux中国 · 10天前 · Git
Git 都 15 岁了，如何入门或学习点新东西
3

专栏 · 白愁高格 · 11天前 · Git
Git常用指令整理
5 1

专栏 · 杭城小刘 · 1月前 · Git / 代码规范

规范化团队 git 提交信息

44 21

专栏 · 不一样的科技宅 · 1月前 · Git

不用代理加速git clone

17 11

专栏 · 以王姓自居 · 6天前 · Git

Git的奇技淫巧🙈