阿里南京技术专刊
2018年05月16日 阅读 49680

关注

优雅的提交你的 Git Commit Message

题注：如果喜欢我们的文章别忘了点击关注阿里南京技术专刊呦~ 本文转载自 阿里南京技术专刊-知乎，欢迎大牛小牛投递阿里南京前端/后端开发等职位，详见 阿里南京诚邀前端小伙伴加入~。

> commit message 是开发的日常操作, 写好 log 不仅有助于他人 review, 还可以有效的输出 CHANGELOG, 对项目的管理实际至关重要, 但是实际工作中却常常被大家忽略. 希望通过本文, 能够帮助大家重视和规范 commit message 的书写.

起因

知乎上有个问题: 如何写好 Git commit log? 很有意思, 能看到各种提交风格: 有用 emoji 的, 有用唐诗的, 有用随机生成的. 风格没有对错, 只要能够体现出 commit 所做的修改即可.

但是最让我印象深刻的是 @李华桥 的答案:

> 这种东西，当然要借助工具了，才能够写得即规范，又格式化，还能够支持后续分析。 目前比较建议的是，使用终端工具 commitizen/cz-cli + commitizen/cz-conventional-changelog + conventional-changelog/standard-version 一步解决提交信息和版本发布。
> 甚至，如果想更狠一点，在持续集成里面加入 marionebl/commitlint 检查 commit 信息是否符合规范，也不是不可以。

本文就顺着这个方向, 给大家介绍下如何保障项目 commit message 的规范和格式化.

Commit Message 格式

目前规范使用较多的是 Angular 团队的规范, 继而衍生了 Conventional Commits specification. 很多工具也是基于此规范, 它的 message 格式如下:

复制代码

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

我们通过 git commit 命令带出的 vim 界面填写的最终结果应该类似如上这个结构, 大致分为三个部分(使用空行分割):

- 标题行: 必填, 描述主要修改类型和内容
- 主题内容: 描述为什么修改, 做了什么样的修改, 以及开发的思路等等
- 页脚注释: 放 Breaking Changes 或 Closed Issues

分别由如下部分构成:

- type: commit 的类型
- feat: 新特性
- fix: 修改问题
- refactor: 代码重构
- docs: 文档修改
- style: 代码格式修改, 注意不是 css 修改
- test: 测试用例修改
- chore: 其他修改, 比如构建流程, 依赖管理.
- scope: commit 影响的范围, 比如: route, component, utils, build...
- subject: commit 的概述, 建议符合 50/72 formatting
- body: commit 具体修改内容, 可以分为多行, 建议符合 50/72 formatting
- footer: 一些备注, 通常是 BREAKING CHANGE 或修复的 bug 的链接.

这样一个符合规范的 commit message, 就好像是一份邮件.

git commit 模板

如果你只是个人的项目, 或者想尝试一下这样的规范格式, 那么你可以为 git 设置 commit template, 每次 git commit 的时候在 vim 中带出, 时刻提醒自己:

修改 ~/.gitconfig, 添加:

复制代码

```
[commit]
template = ~/.gitmessage
```


目录

新建 ~/.gitmessage 内容可以如下:

复制代码

```
# head: <type>(<scope>): <subject>
# - type: feat, fix, docs, style, refactor, test, chore
# - scope: can be empty (eg. if the change is a global or difficult to assign to a single compone
# - subject: start with verb (such as 'change'), 50-character line
#
# body: 72-character wrapped. This should answer:
# * Why was this change necessary?
# * How does it address the problem?
# * Are there any side effects?
#
# footer:
# - Include a link to the ticket, if any.
# - BREAKING CHANGE
#
```

Commitizen: 替代你的 git commit

我们的目标还是要通过工具生成和约束, 那么现在就开始吧.

commitizen/cz-cli, 我们需要借助它提供的 git cz 命令替代我们的 git commit 命令, 帮助我们生成符合规范的 commit message.

除此之外, 我们还需要为 commitizen 指定一个 Adapter 比如: cz-conventional-changelog (一个符合 Angular团队规范的 preset). 使得 commitizen 按照我们指定的规范帮助我们生成 commit message.

全局安装

复制代码

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

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

项目级安装

复制代码

```
npm install -D commitizen cz-conventional-changelog
```

package.json中配置:

复制代码

```
"script": {
    ...,
    "commit": "git-cz",
},
 "config": {
    "commitizen": {
      "path": "node_modules/cz-conventional-changelog"
    }
  }
```

如果全局安装过 commitizen, 那么在对应的项目中执行 git cz or npm run commit 都可以.

效果如下:

```
➜ ng-poopy  master ✗  git add .
➜ ng-poopy  master ✗  git cz

All commit message lines will be cropped at 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, formatting, missing semi-colons, etc)
  refactor: A code change that neither fixes a bug or adds a feature
  perf:     A code change that improves performance
  test:     Adding missing tests
  chore:    Changes to the build process or auxiliary tools
            and libraries such as documentation generation
```

自定义 Adapter

也许 Angular 的那套规范我们不习惯, 那么可以通过指定 Adapter cz-customizable 指定一套符合自己团队的规范.

全局 或 项目级别安装:

复制代码

```
npm i -g cz-customizable
or
npm i -D cz-customizable
```

修改 .czrc 或 package.json 中的 config 为:

复制代码

```
{ "path": "cz-customizable" }
or
  "config": {
    "commitizen": {
      "path": "node_modules/cz-customizable"
    }
  }
```

同时在~/ 或项目目录下创建 .cz-config.js 文件, 维护你想要的格式: 比如我的配置文件: leohxj/.cz-config

效果如下:

Commitlint: 校验你的 message

commitlint: 可以帮助我们 lint commit messages, 如果我们提交的不符合指向的规范, 直接拒绝提交, 比较狠.

同样的, 它也需要一份校验的配置, 这里推荐 @commitlint/config-conventional (符合 Angular团队规范).

安装:

```
复制代码
npm i -D @commitlint/config-conventional @commitlint/cli
```

同时需要在项目目录下创建配置文件 .commitlintrc.js, 写入:

```
复制代码
module.exports = {
  extends: [
    ''@commitlint/config-conventional''
  ],
  rules: {
  }
};
```

针对自定义的 Adapter 进行 Lint

如果你像我一样, 使用的是自定义的 commitizen adapter, 那么你需要:

```
复制代码
npm i -D commitlint-config-cz @commitlint/cli
```

.commitlintrc.js 中写入:

```
复制代码
module.exports = {
  extends: [
    'cz'
  ],
  rules: {
  }
};
```

结合 Husky

校验 commit message 的最佳方式是结合 git hook, 所以需要配合 Husky.

```
复制代码
npm i husky@next
```

package.json 中添加:

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

效果如下:

standard-version: 自动生成 CHANGELOG

通过以上工具的帮助, 我们的工程 commit message 应该是符合 Angular团队那套, 这样也便于我们借助 standard-version 这样的工具, 自动生成 CHANGELOG, 甚至是 语义化的版本号(Semantic Version).

安装使用:

```
复制代码
npm i -S standard-version
```

package.json 配置:

```
复制代码
```

```
"scirpt": {
    ...,
    "release": "standard-version"
}
```

PS: standard-version 有很多其他的特性, 这里不过多涉及, 有兴趣的同学自行尝试.

最后

commit message 的规范性很重要, 但是是否需要像本文这样强制限制, 每个团队和个人都有自己的想法, 但是个人认为: 好的习惯, 受益终身.

> **关注下面的标签，发现更多相似文章**

Angular.js Git JavaScript 前端

评论

输入评论...

Emac 架构师 @ XR
实现了一个简单的CLI工具，用于帮助输入Angular git commit message规范所需的主要字段。欢迎Fork。
github.com/emac/gitx
5天前 回复

幸福就是不断的经历 NA
Git Commit Template
5月前 回复

左手画龙
mark
5月前 回复

Reference_Error FE @首席摸鱼官
各位，菜鸟问一句，这个echo '{ "path": "cz-conventional-changelog" }' > ~/.czrc这句话在哪儿执行？？？ 😂，我执行提示系统找不到指定路径，用的windows的cmd
5月前 回复

hello-acuario
回复 Reference_Error: windows 当然没有这个目录了，在计算机里添加一个path路径
3月前 回复

仑 前端搬砖党
我把我之前的环境变量全部覆盖了，我的天，能不能用>> ~/.czrc
5月前 回复

仑 前端搬砖党
回复 仑: 我以为添加到环境变量，我自己的问题
5月前 回复

一只狗狗 发呆工程师
/usr/local/lib/node_modules/commitizen/dist/commitizen/commit.js:545
  (0, _git.commit)(sh, repoPath, template, { ...options,
                                             ^^^...
展开
7月前 回复

查看更多 >

相关推荐

专栏 · 徐小夕 · 3小时前 · React.js / JavaScript
基于react搭建一个通用的表单管理配置平台（vue同）
28 1

专栏 · *5102 · 1天前 · JavaScript
JS 万字总结 重量级干货！！！
274 37

荐 · 专栏 · 前端劝退师 · 3天前 · 前端
抄笔记：尤雨溪在Vue3.0 Beta直播里聊到了这些...
1189 166

专栏 · 前端早早聊 · 1天前 · 前端
精神小伙，如何让自己成为前端 TeamLeader 候选人
143 8

荐 · 专栏 · 掘金酱 · 23小时前 · 面试 / 前端
【新鲜面经】技术面试并不难，掘友经验带上岸 | 掘金技术征文展（第二弹）
65 21

专栏 · 前端早早聊 · 2天前 · 前端 / 强化学习
哇哦！高级前端技术专家的职业规划长这样儿

241 25

专栏 · 前端小智 · 1天前 · JavaScript

web 图像技术：前端引入图片的各种方式及其优缺点

78 2

专栏 · 小又又 · 1天前 · 前端

转行学前端的第 17 天：基础店铺页面结构确认

19 31

专栏 · 前端早早聊 · 2天前 · 前端 / 强化学习

前端技术专家(P8)的规划能力如何训练，答案全给你

159 12

专栏 · 前端小时 · 3天前 · 前端

前端人应该知道的网站和工具

149 23

专栏 · 好学习吧、 · 1天前 · 前端

搞定混合开发面试，这一篇就够了！

20