人工智障研究中心 PRO
2019-04-25 15:51:11 · 阅读 · 2018

关注

Git commit message 规范

git是现代软件工程的记录和审核工具，可用来对你的commit message进行大范围的规范，这些工具可用于自动生成版本说明（release notes）或自动生成版本号（semver）。

用什么规范？

现在市面上比较流行的方案有 约定式提交规范（Conventional Commits），它受到了 angular团队规范 的启发，并在第 4 次版本上以MIT协议。约定式提交规范 是一种基于提交消息的轻量级约定。它提供了一组简单易懂的规则，这种规则有助于在提交历史记录上建立清晰的意图，这不仅有助于 Semver 自动化，在提交信息中表明此新特性，bug 修复和/或不稳定更改。约定 message 格式如下：

```
<类型>(可选的范围): <描述>

[可选的正文]

[可选的脚注]
```

Quick Start

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

commitizen 是一个撰写有合格 commit message 的工具，用于代替 git commit 指令。而 cz-conventional-changelog 是 commitizen 的 conventional changelog标准（约定式提交规范）。基于不同需求，也可以使用不同适配器。

```
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 ，

```
npm install --save-dev husky
```

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

```
npx husky install
```

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

commitlint 只能提供校验规则，无法触发内核，如下内容仅最近的官方commitlint文档。

3. 添加package配置

创建 commitlint.config.js

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

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

引入 husky ：

```
# package.json

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

4. 使用

执行 git cz 进入interactive模式，根据提示填写

```
1.Select the type of change that you're committing :选择问题类型 (Type)
2.What is the scope of this change (e.g. component or file name): 填写问题范围 (scope)
3.Write a short, imperative tense description of the change: 写一个简短的描述 (subject)
4.Provide a longer description of the change: (press enter to skip) 对于问题的一些补充说明 (body)
5.Are there any breaking changes? (y/n) 是否有破坏性修改? 默认 (n)(enter)
6.Does this change affect any open issues? (y/n) 问题是否影响了问题? 默认 (n)(enter)
```

生成的 commit message格式如下：

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

填写规则后，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 都适用于 git cz 指令之，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，我们对此规范做了 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五个常用type，其余type 较不常用。

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

# 特殊type
docs:     文档改动
style:    不影响代码含义的改动，例如去掉空格、删除分号
build:    构建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打回的commit

# 较不常用type
test:     添加测试或者修改测试
perf:     提高性能的改动
ci:       对ci（持续集成软件）有关的改动
chore:    其他不一定适合放在以上类别的其他改动或构建工具的改动
```

当一次改动既属于 fix type 又 feat type 时，统一采用 主要type 。

2. scope

scope 也为必填项，用于描述此改动范围，格式为项目名/模块名，例如： node-polymer 项目的bookshelf 模块 。scope 不要加空格以示，如果一次commit涉及多个模块，请以斜线分隔多次commit，以便更好追踪和编产。

3. body

body 填写详细描述，主要描述 具体注意修改点 及 修改动机 ，对于小改动无不可要求，但是重大重要，更要详细说明改动内容和原因。

4. break changes

break changes 指明是否为了破坏性修改，详见break changes的定义说明编辑状态，例如国际化，affect issues 指明是否影响一个或多个问题，过往等等。

5. affect issues

affect issues 指明是否影响了某个问题，例如因为使用bug修复，则在 commit message 中可以填写类似fix #1，若要开始或结束问题跟踪的工具 jira 与 gitlab，参考文档：docs.github.com/en/issues/...

填写方式如如：

```
ref #1234,#5
Fix #1234,#5
```

示例

- 完整的commit message示例

```
$ cz-cli -T -S -n cz-conventional-changelog -r 1

? Select the type of change that you're committing: (Press <space> to select, <j> and <k> to navigate, <enter> to confirm)
  feat:  A new feature
  fix:   A bug fix
  docs:  Only documentation changes
  style: Changes that do not affect the meaning of the code (white-space, formatting, missing semicolons, 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
  chore: Changes to the build process or auxiliary tools and libraries such as documentation generation
? What is the scope of this change (e.g. component or file name): (press enter to skip)
? Write a short, imperative tense description of the change (max 83 chars):
 (nyc) add .gitignore
? Provide a longer description of the change: (press enter to skip)
? Are there any breaking changes? (y/n) n
? Does this change affect any open issues? (y/n) n
? Write a message for this commit [Preview]: feat(): (nyc) add .gitignore
? Choose a remote to push to origin
? Find more information at https://github.com/commitizen/cz-cli/issues/ when you encounter any issues.
? Are you sure you want to proceed with the commit? (Y/n) Y
```