团队代码规范标准

commit 规范

commitlint 可以帮助你的团队使用统一的 commit 规范。

使用

install

```
npm install @commitlint/{cli,config-conventional} -D
```

在 **git commit** 提交时进行 **commit** 检查是否规范，因此需要对 **commit** 进行拦截，只有符合相应的格式才能提交代码。社区中可以通过 **Husky** 来完成这件事情。

Install husky

```
# Install
npm install husky --save-dev

# Activate hooks
npx husky install
```

Add hook

```
npx husky add .husky/commit-msg  'npx --no -- commitlint --edit ${1}'
```

当前根目录下新建 commitlint.config.js

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

我们一般直接使用 **@commitlint/config-conventional** 提供的规范即可，它所规定的 commit 信息格式一般如下：

```
git commit -m <type>[optional scope]: <description>
```

常用 type 类别

type：用于表明这次提交代码的改动类型，常用的类型如下：

- feat：新增功能
- fix：bug 修复
- docs：文档更新
- style：不影响程序逻辑更改（例如格式化，分号不全等）
- refactor：代码重构（既不新增功能也不是 bug 修复）
- perf：性能优化
- test：新增测试用例或更改已有测试用例
- build：影响构建系统或外部依赖关系的更改（例如gulp、webpack、npm等）
- ci：CI 配置文件与脚本的修改（例如：Travis、Circle等）
- chore：不涉及到修改 src 文件或测试文件的其他更改
- revert：回退到之前的commit提交记录

optional scope

一个可选的修改范围。用于标识此次提交主要涉及到代码中哪个模块的改动。

description

用于描述此次提交的内容。

示例

```
git commit -m 'feat(xx模块): 增加 xx 功能'
```

参考资料

1. https://commitlint.js.org/#/

vite 相关

1. GLob 模式
2. 预构建

依赖预构建

1. 将其他格式（如CommonJS、UMD）的产物转化为 ESM 格式，使得其在浏览器中通过