统一代码规范，使代码更容易阅读
统一提交规范，更好的提高工作效率
有效的输出CHANGELOG 保持对版本的跟踪
工具化，更容易实施和保证结果

HUSKY 规范配置

```
npm i --save husky lint-staged commitlint
```

配置

```
"scripts": {
    "lint": "vue-cli-service lint"
  },
// 这种写法 vue 可以被支持
//"gitHooks": {
//   "pre-commit": "lint-staged"
// },
"husky": {
    "hooks": {
      "pre-commit": "lint-staged",
      "commit-msg": "commitlint -E HUSKY_GIT_PARAMS"
    }
  },
  "lint-staged": {
    "*.{js,vue}": [
      "vue-cli-service lint",
      "git add ."
    ]
  }
```

配置文件

```
// commitlint.config.js
module.exports = {
  extends: ["@commitlint/config-conventional"],
  rules: {
    "type-enum": [
      2,
      "always",
```

```
      [
        "feat",
        "build",
        "fix",
        "docs",
        "style",
        "refactor",
        "test",
        "chore",
        "revert"
      ]
    ],
    "subject-full-stop": [0, "never"],
    "subject-case": [0, "never"]
  }
};
```

使用

```
git add .
git commit -m 'feat:  新增测试' //  保持与commitlint.config 一致
```

GIT-CZ 规范配置

应用工具介绍

> eslint
> 静态检查工具

> lint-staged
> 是一个在git暂存文件上运行linters的工具，Lint-staged仅仅是文件过滤器，对该文件执行相应的动作，主要是提高了 文件检查的效率

> git-cz
> 使用 Commitizen 提交时，系统会提示您在提交时填写任何必需的提交字段。不再等到以后运行 git 提交挂钩并拒绝提交.

> cz-conventional-changelog
> 规范提交信息

> standard-version
>
> - 基于现有信息及standard-version命令参数生成新的version版本号
> - 创建或者修改CHANGELOG.md
> - 基于以上修改创建git commit
> - 使用版本号创建git tag

```
// 安装
npm i --save cz-conventional-changelog git-cz lint-staged standard-version
```

配置

package.json

```
"scripts": {
    "lint": "vue-cli-service lint",
    "changelog": "standard-version",  // 每次执行会根据 "version": "2.3.0", 字段 最后一位增加 1
    "commit": "git add . && npx git-cz"
},
"config": {
    "commitizen": {
      "path": "./node_modules/git-cz"
    }
  },
  "gitHooks": {
    "pre-commit": "lint-staged",
    "commit-msg": "node verifyCommitMsg.js"
  },
  "lint-staged": {
    "*.{js,vue}": [
      "vue-cli-service lint",
      "git add ."
    ]
  }
```

新增文件 verifyCommitMsg.js

```
const chalk = require('chalk')  // eslint-disable-line
const msgPath = process.env.GIT_PARAMS;
const msg = require("fs")
  .readFileSync(msgPath, "utf-8")
  .trim();

const commitRE = /^(v\d+\.\d+\.\d+(-(alpha|beta|rc.\d+))?$)|((revert:
)?
(feat|fix|docs|style|refactor|perf|test|workflow|ci|chore|types|build)
(\(.+\))?: .{1,50})/;
const mergeRE = /Merge .{1,50}/;
if (!commitRE.test(msg) && !mergeRE.test(msg)) {
  console.log();
  console.error(
    `  ${chalk.bgRed.white(" ERROR ")} ${chalk.red(
      `Commit message 格式不正确.`
    )}\n\n` +
      chalk.red(`  本项目强制使用Commit message 规范格式. 如:\n\n`) +
      `    ${chalk.green(`feat(compiler): add 'comments' option`)}\n` +
      `    ${chalk.green(
        `fix(v-model): handle events on blur (close #28)`
      )}\n\n` +
      chalk.red(
        `  请使用 ${chalk.cyan(`npm run commit`)} 命令生成Commit
message.\n`
      )
  );
  process.exit(1);
}
```

新增文件 verifyCommitMsg.js

```
module.exports = {
  list: [
    "test",
    "feat",
    "fix",
    "chore",
    "docs",
    "refactor",
    "style",
    "ci",
    "perf"
  ],
  maxMessageLength: 64,
  minMessageLength: 3,
  questions: [
    "type",
    "scope",
    "subject",
    // 'body',
```

```
    // 'breaking',
    // 'issues',
    "lerna"
  ],
  scopes: [],
  types: {
    chore: {
      description: "构建流程或工具脚本等变更",
      emoji: "🤖",
      value: "chore"
    },
    ci: {
      description: "CI 相关的变更",
      emoji: "🎡",
      value: "ci"
    },
    docs: {
      description: "只包含文档变更",
      emoji: "✏️",
      value: "docs"
    },
    feat: {
      description: "新功能",
      emoji: "🎸",
      value: "feat"
    },
    fix: {
      description: "Bug修复",
      emoji: "🐛",
      value: "fix"
    },
    perf: {
      description: "提升性能的变更",
      emoji: "⚡️",
      value: "perf"
    },
    refactor: {
      description: "不影响功能和bug的重构",
      emoji: "💡",
      value: "refactor"
    },
    release: {
      description: "版本发布",
      emoji: "🏹",
      value: "release"
    },
    style: {
      description: "代码格式化、注释等变更",
      emoji: "💄",
      value: "style"
    },
    test: {
```

```
      description: "添加测试用例",
      emoji: "💍",
      value: "test"
    }
  }
};
```

使用

问题

经过测试发现只是更改eslint的格式，提交会出现问题；暂无找到解决办法

```
✖ Prevented an empty git commit!
```

这种情况采用

```
git add .
git commit -m 'style: xxx' // 格式可参考 changelog.config 格式要求
// 可以正常提交
```