

webstorm的Eslint环境配置&husky配置

ESLint是一个插件化的javascript代码检测工具，它可以用于检查常见的JavaScript代码错误，也可以进行代码风格检查，我们可以根据自己的喜好指定一套ESLint配置，然后应用到所编写的项目上，从而实现辅助编码规范的执行，有效控制项目代码的质量。

webstorm的Eslint环境配置

1. 安装eslint

首先通过npm安装eslint，通过以下命令全局安装

```
npm install eslint -g
```

2. 添加.eslintrc文件

创建一个名为.eslintrc的js文件在项目根目录下
该文件用于定义一些规则，例如添加以下内容

```
module.exports = {
  "env": {
    "browser": true,
    "node": true
  },
  "extends": "eslint:recommended",
  "rules": {
    "indent": [
      2,
      2,
      ],
    "linebreak-style": [
      2,
      "unix"
    ],
    "quotes": [
      2,
      "double"
    ],
    "semi": [
      2,
      "always"
    ],
    "camelcase": 2, //
  },
  "space-before-blocks": "error", //
  "space-in-parens": [ //
    2,
    "always"
  ],
  "space-infix-ops": "error", //
  "valid-jsdoc": [ // JSDoc
    2,
    ],
  //
  // @arg@argument @param
  //
  // @return @returns
  //
  //
  "require-jsdoc": ["error", {
    "require": {
      "FunctionDeclaration": true,
      "MethodDefinition": true,
```

```

    "ClassDeclaration": true,
    "ArrowFunctionExpression": true
  }
}],
  "no-undef": 2, // /*global */
  "no-unused-vars": 2, //
  "no-redeclare": 2, //
  "max-nested-callbacks": [ //
    2,
    4
  ],
  "max-params": [ //
    2,
    3
  ],
  "max-statements": [ //
    2,
    200
  ],
  "max-statements-per-line": [ //
    2,
    { "max": 1 }
  ],
  "max-lines": [ //
    2, 1200
  ],
  "new-cap": 2, //
  "array-element-newline": [ //
    2,
    { "multiline": true }
  ],
  "keyword-spacing": [ //
    2,
    { "before": true }
  ],
  //
  "comma-dangle": [
    2, "never"
  ], //
  "no-debugger": 2, //debugger
  "no-constant-condition": 2, //
  "no-dupe-args": 2, //
  "no-dupe-keys": 2, //
  "no-duplicate-case": 2, //case
  "no-empty-character-class": 2, //
  "no-invalid-regexp": 2, //
  "no-func-assign": 2, //
  "valid-typeof": 2, //
  "no-unreachable": 2, //
  "no-unexpected-multiline": 2, //
  "no-sparse-arrays": 2, //
  "no-shadow-restricted-names": 2, //
  "no-cond-assign": 2, //
  "no-native-reassign": 2, //
  }
};

```

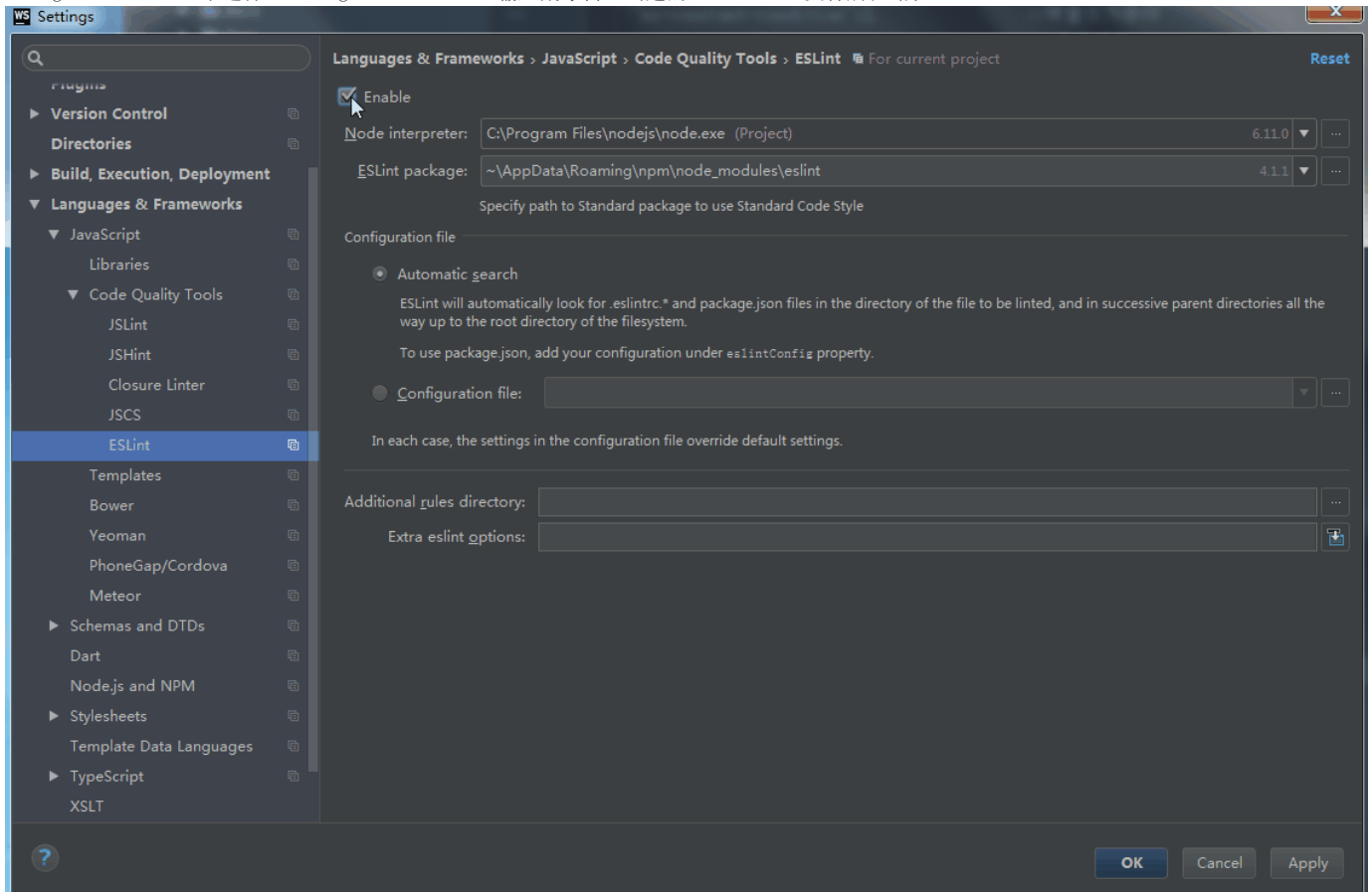
3. 配置webstorm

为了在编辑器里用上eslint我们需要做以下配置

在setting>Language&frameworks>javascript>code quality tools>ESLint 中选中enable

Eslint package: eslint的安装目录

configuration file 中选择 configuration file 输入刚才自己创建的 .eslintrc文件所在路径

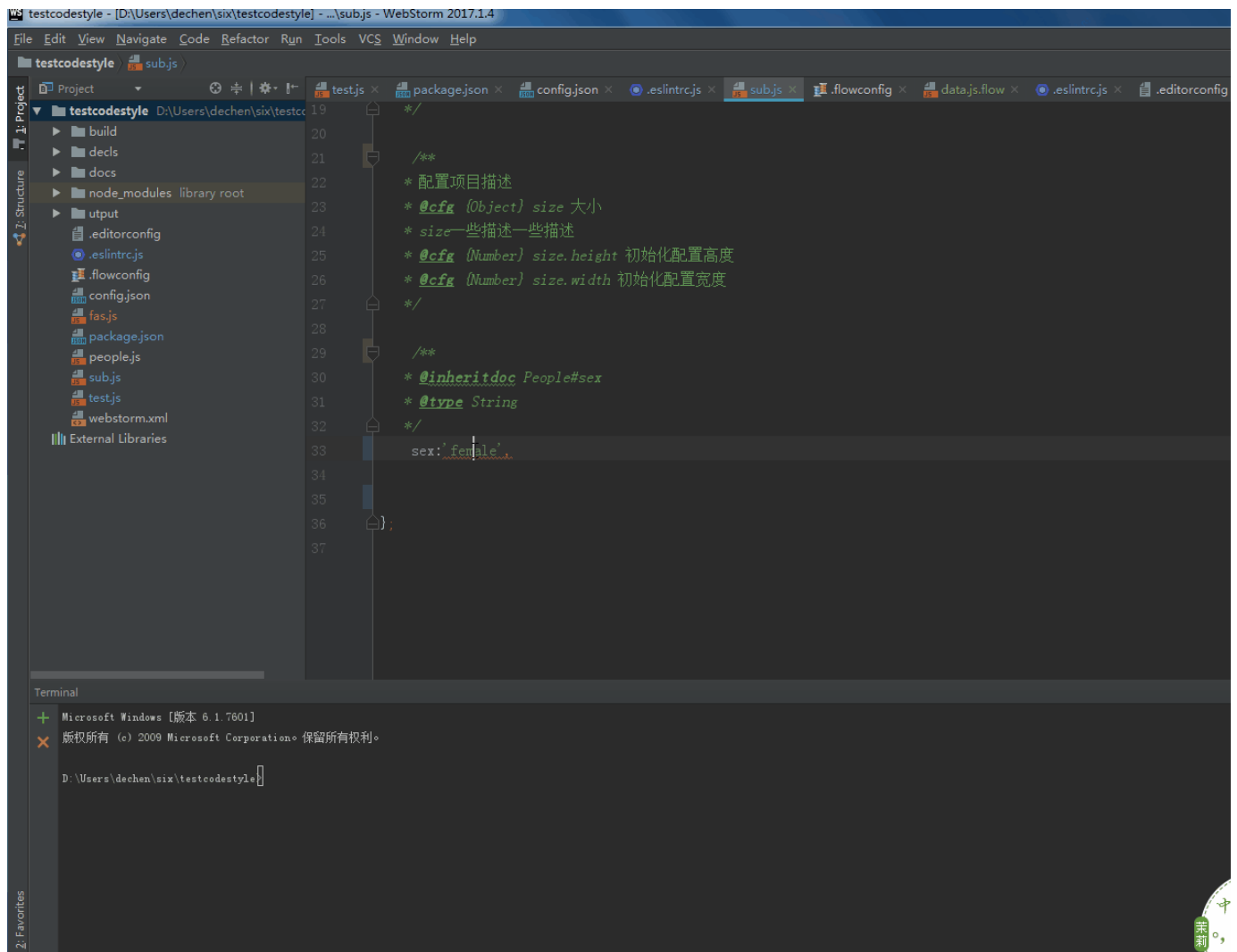


ESLint的使用

1. 编辑器中

- 提示

直接在编辑器中可以看见不符合eslint规范的代码会被标注出来
例如：



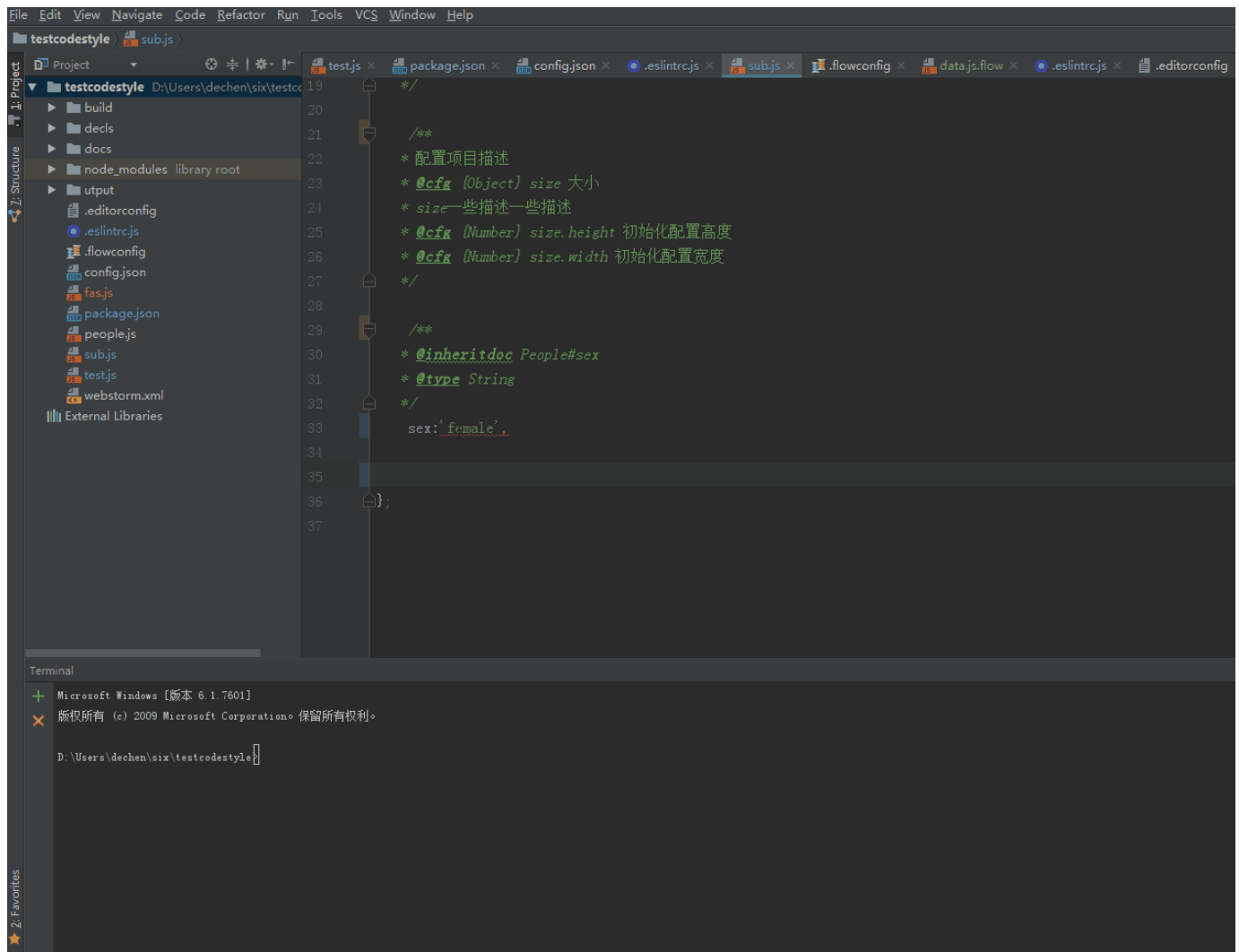
- 修复

右键可以看到[Fix ESLint Problems](#)

点击进行修复

部分问题可以通过Fix ESLint Problems自动修复

但是只能修复部分问题，剩下的只能自己手动修复



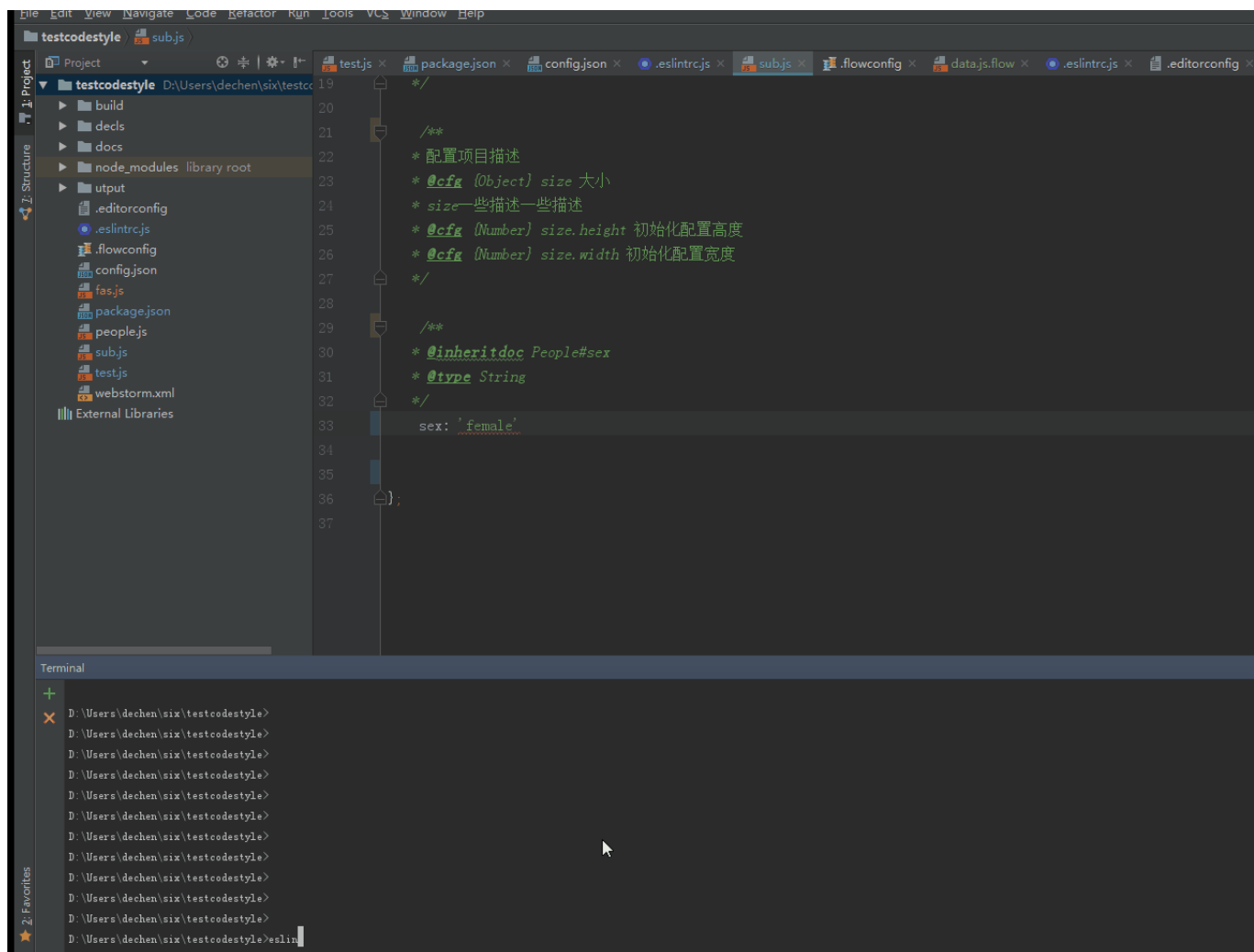
2. Terminal中

- 检查

在Terminal中输入

`eslint XXX.js`

xxx.js为需要进行eslint检查的文件



- 修复

在Terminal中输入

```
eslint xxx.js fix
```

xxx.js 为你要修复的文件

同样，只能修复部分问题，剩下的只能自己手动修复

参考

[eslint中文文档](#)

使用husky在代码提交前进行自动eslint检查

虽然我们已经使用了eslint进行代码检查，但是我们可以忽略eslint的检查结果，直接commit代码，我们需要避免这种情况。

husky这个插件能够在git commit之前进行自动化处理，在这里我们使用husky自动执行eslint脚本，进行代码检查，也就是说只有通过了eslint检查才可以commit代码。

husky配置

1. 安装

安装husky

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

2. 编辑package.json

安装完成husky后，我们在package.json中添加以下内容

```
"scripts": {
  "codecheck": "eslint test.js",
  "precommit": "npm run codecheck -q"
}
```

配置了precommit之后每次代码提交之前git都会自动去跑precommit中的脚本

- 这里我们把precommit设置为 `npm run codecheck -q`
- 其中的codecheck设置为 `"codecheck": "eslint test.js"`
- test.js 改成需要进行eslint检查的文件或目录

husky会去读取package.json中的配置信息

husky使用

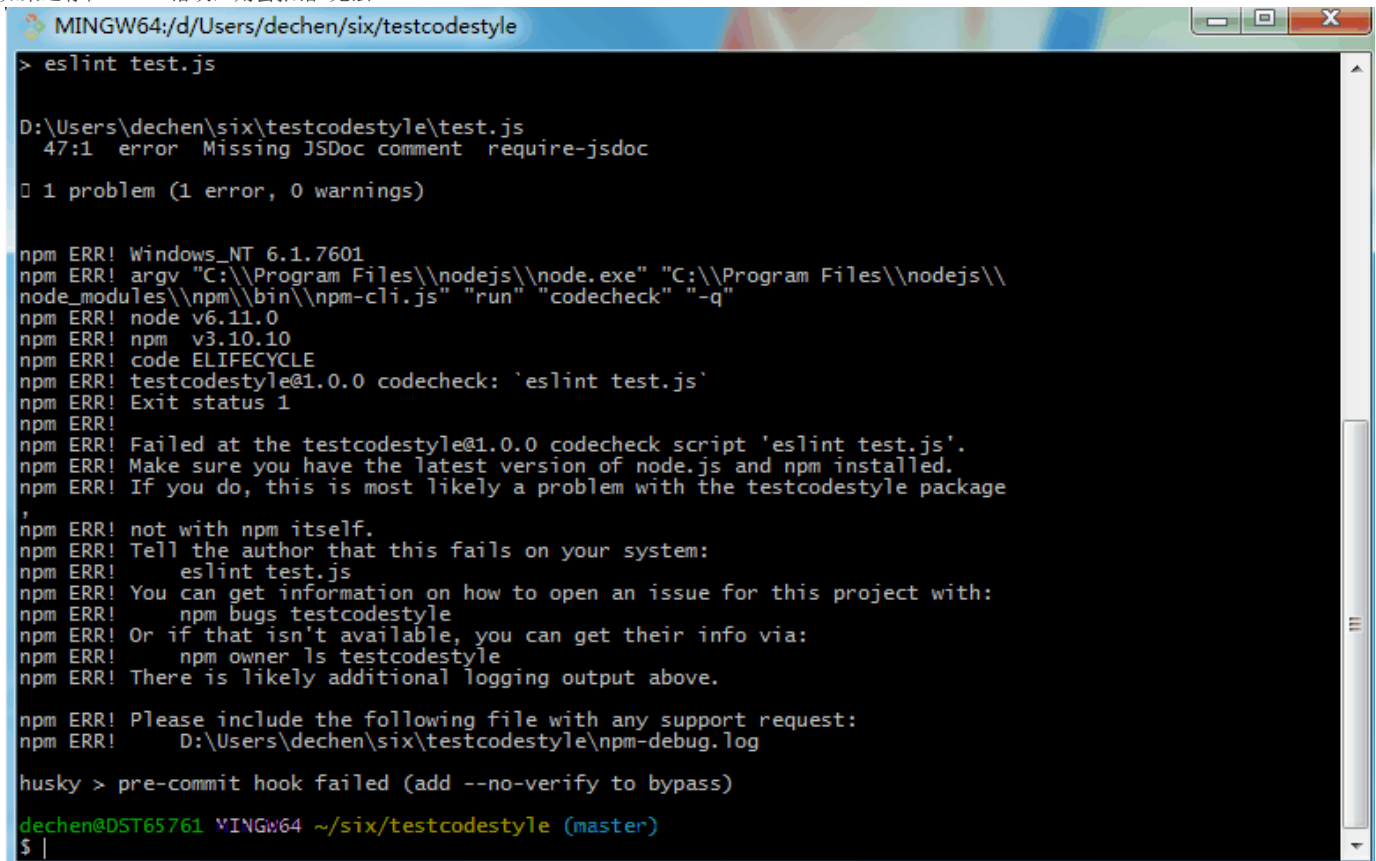
配置完了husky之后

尝试commit

```
git commit -m "....."
```

会发现在commit之前会自动进行eslint检查，只有检查通过了才可以成功提交

如果还存在ealint错误，则会报错 无法commit



```
MINGW64/d:/Users/dechen/six/testcodestyle
> eslint test.js

D:\Users\dechen\six\testcodestyle\test.js
  47:1  error  Missing JSDoc comment  require-jsdoc

✖ 1 problem (1 error, 0 warnings)

npm ERR! Windows_NT 6.1.7601
npm ERR! argv "C:\\Program Files\\nodejs\\node.exe" "C:\\Program Files\\nodejs\\
node_modules\\npm\\bin\\npm-cli.js" "run" "codecheck" "-q"
npm ERR! node v6.11.0
npm ERR! npm v3.10.10
npm ERR! code ELIFECYCLE
npm ERR! testcodestyle@1.0.0 codecheck: `eslint test.js`
npm ERR! Exit status 1
npm ERR!
npm ERR! Failed at the testcodestyle@1.0.0 codecheck script 'eslint test.js'.
npm ERR! Make sure you have the latest version of node.js and npm installed.
npm ERR! If you do, this is most likely a problem with the testcodestyle package
,
npm ERR! not with npm itself.
npm ERR! Tell the author that this fails on your system:
npm ERR!   eslint test.js
npm ERR! You can get information on how to open an issue for this project with:
npm ERR!   npm bugs testcodestyle
npm ERR! Or if that isn't available, you can get their info via:
npm ERR!   npm owner ls testcodestyle
npm ERR! There is likely additional logging output above.

npm ERR! Please include the following file with any support request:
npm ERR!   D:\Users\dechen\six\testcodestyle\npm-debug.log

husky > pre-commit hook failed (add --no-verify to bypass)

dechen@DST65761 MINGW64 ~/six/testcodestyle (master)
$ |
```

参考

husky github仓库

解释husky