浅析ESLint

范洋

内容概括

Eslint 的主要功能介绍

现场给我们的 za-crm-pc项目配置eslint

ESlint简介

在团队协作中，为避免低级 Bug、产出风格统一的代码，会预先制定编码规范。
ESlint是一款流行的代码风格检测工具，通过预定规则对代码进行检测，能够有效控制代码质量。

ESLint最初是由Nicholas C. Zakas于2013年6月创建的开源项目。
它的目标是提供一个插件化的javascript代码检测工具。
它被设计为易于拓展的，具有大量的自定义规则，并且很容易通过插件的形式来安装。成为了目前前端较为主流的代码质量检测工具。

Eslint 的基础部分梳理

配置
规则
格式化
集成
进阶

1. 配置方式

```
fanyangdeMacBook-Pro:test fanyang$
fanyangdeMacBook-Pro:test fanyang$ ls
package.json    src
fanyangdeMacBook-Pro:test fanyang$ eslint --init
? How would you like to configure ESLint? Inspect your JavaScript file(s)
? Which file(s), path(s), or glob(s) should be examined? src
? What format do you want your config file to be in? JSON
? Which version of ECMAScript do you use? ES2018
? Are you using ES6 modules? Yes
? Where will your code run? Browser, Node
? Do you use CommonJS? Yes
? Do you use JSX? No
Determining Config: 100% [==============================] 0.2s elapsed, eta 0.0s


Enabled 262 out of 264 rules based on 1 file.
Local ESLint installation not found.
The config that you've selected requires the following dependencies:

eslint@latest
Successfully created .eslintrc.json file in /Users/fanyang/D/ppt/test
ESLint was installed locally. We recommend using this local copy instead of your globally-installed copy.
fanyangdeMacBook-Pro:test fanyang$ ls -al
total 32
drwxr-xr-x  5 fanyang  staff   160  9  4 23:26 .
drwxr-xr-x  4 fanyang  staff   128  9  4 23:23 ..
-rw-r--r--  1 fanyang  staff  8218  9  4 23:26 .eslintrc.json
-rw-r--r--  1 fanyang  staff   200  9  4 23:26 package.json
drwxr-xr-x  3 fanyang  staff    96  9  4 23:25 src
```

可以通过以下三种方式配置 ESLint:

- 使用 `.eslintrc` 文件（支持 JSON 和 YAML 两种语法）；
- 在 `package.json` 中添加 `eslintConfig` 配置块；
- 直接在代码文件中定义。

.eslintrc 文件示例

```
{
    "env": {
        "browser": true,
        "commonjs": true,
        "es6": true,
        "node": true
    },
    "extends": ["plugin:vue/base"],
    "parserOptions": {
        "ecmaVersion": 2018,
        "sourceType": "module"
    },
    "rules": {
        "indent": ["error", "tab",{"ImportDeclaration":"off"}],
        "quotes": ["warn", "single"],
        "semi": ["warn", "never"]
    }
}
```

.package.json 文件示例

```
{
  "name": "mypackage",
  "version": "0.0.1",
  "eslintConfig": {
    "env": {
      "browser": true,
      "node": true
    }
  }
}
```

文件内配置

```
/* eslint-disable */
var obj = { key: 'value', }; // I don't care about IE8
/* eslint-enable */
```

简略的看一看主要配置项

```
{
    //extends用于引入某配置作为基础配置，然后再在后续的rules中对其进行扩展
    "extends": "eslint:recommended",
    //如果你想使用插件中的环境变量，请先把插件名写入"plugins"数组中，然后再在"env":{}中以
    //"插件名/插件中的需引入的环境名"的方式进行指定。
    "plugins": ["example"],
    "env": {
        //An environment defines global variables that are predefined.
        //定义env会带进来一些全局变量
        "browser": true,
        "node": true,
        "es6":true,
        "mocha":"true",
        "qunit":true,
        "jquery":true,
        "mongo":true,
        //通过插件名命名空间引入插件中的环境
        "example/custom": true
    },
    "globals": {
        // globals are the additional global variables your script accesses during execution.
        // 即插件在执行过程中用到的其它全局变量
        "angular": true,
    },
    "rules": {
        //which rules are enabled and at what error level.
        //这里指定用哪些规则进行eslint检查以及每个规则的错误级别：0或者off表示规则关闭，
        //出错也被忽略；1或者warn表示如果出错会给出警告；2或者error表示如果出错会报出错误
        "semi": ["error", "always"],
        "quotes": ["error", "double"]
    },
    //parser指定解析器，默认的为espree，可选的还有Esprima、Babel-ESLint。
    //babel-eslint is a wrapper around the Babel parser that makes it compatible with ESLint.
        //babel-eslint是一个Babel parser的包装器，这个包装器使得Babel parser可以和ESLint协调工作
    "parser": "babel-eslint",
    // 解析器选项
    "parserOptions": {
        //ecmaVersion指定ECMAScript的版本，可选值有3\5\6\7，默认是5
       "ecmaVersion": 6,
       //sourceType指定被检查的文件是什么扩展名的，可选项"script"和"module"，默认是"script"。"module"是ES6的。
       "sourceType": "module",
       //ecmaFeatures指定你想使用哪些额外的语言特性
       "ecmaFeatures": {
           "jsx": true
       }
    },
}
```

2 .规则

规则

所有的规则默认都是禁用的

在配置文件中，使用 "extends": "eslint:recommended" 来启用推荐的规则　规则：http://eslint.cn/docs/rules/

ESLint 附带有大量的规则。你可以使用注释或配置文件修改你项目中要使用的规则。要改变一个规则设置，你必须将规则 ID 设置为下列值之一：

"off" 或 0 - 关闭规则
"warn" 或 1 - 开启规则，使用警告级别的错误：warn (不会导致程序退出)
“error" 或 2 - 开启规则，使用错误级别的错误：error (当被触发的时候，程序会退出)

注释方式使用

```
/* eslint eqeqeq: "off", curly: "error" */
```

配置文件形式

```
{
    "rules": {
        "eqeqeq": "off",
        "curly": "error",
        "quotes": ["error", "double"]
    }
}
```

规则示例

命令行的 `--fix` 选项用来自动修复规则所报告的问题（目前，大部分是对空白的修复），在下文中会有一个的扳手图标

Possible Errors

这些规则与 JavaScript 代码中可能的错误或逻辑错误有关：

		for-direction	强制 “for” 循环中更新子句的计数器朝着正确的方向移动
		getter-return	强制 getter 函数中出现 `return` 语句
		no-await-in-loop	禁止在循环中出现 `await`
✔		no-compare-neg-zero	禁止与 -0 进行比较
✔		no-cond-assign	禁止条件表达式中出现赋值操作符
✔		no-console	禁用 `console`
✔		no-constant-condition	禁止在条件中使用常量表达式
✔		no-control-regex	禁止在正则表达式中使用控制字符
✔	🔧	no-debugger	禁用 `debugger`
✔		no-dupe-args	禁止 `function` 定义中出现重名参数
✔		no-dupe-keys	禁止对象字面量中出现重复的 key
✔		no-duplicate-case	禁止出现重复的 case 标签
✔		no-empty	禁止出现空语句块
✔		no-empty-character-class	禁止在正则表达式中使用空字符集
✔		no-ex-assign	禁止对 `catch` 子句的参数重新赋值
✔	🔧	no-extra-boolean-cast	禁止不必要的布尔转换
	🔧	no-extra-parens	禁止不必要的括号
✔	🔧	no-extra-semi	禁止不必要的分号
✔		no-func-assign	禁止对 `function` 声明重新赋值
✔		no-inner-declarations	禁止在嵌套的块中出现变量声明或 `function` 声明
✔		no-invalid-regexp	禁止 `RegExp` 构造函数中存在无效的正则表达式字符串
✔		no-irregular-whitespace	禁止在字符串和注释之外不规则的空白
✔		no-obj-calls	禁止把全局对象作为函数调用
		no-prototype-builtins	禁止直接调用 `Object.prototypes` 的内置属性
✔	🔧	no-regex-spaces	禁止正则表达式字面量中出现多个空格
✔		no-sparse-arrays	禁用稀疏数组
		no-template-curly-in-string	禁止在常规字符串中出现模板字面量占位符语法
✔		no-unexpected-multiline	禁止出现令人困惑的多行表达式
✔		no-unreachable	禁止在`return`、`throw`、`continue` 和 `break` 语句之后出现不可达代码

使用自定义规则

以 eslint-plugin-react 为例，安装以后，需要在 ESLint 配置中开启插件，其中 `eslint-plugin-` 前缀可以省略

```
{
  "plugins": [
      "react"
  ]
}
```

接下来开启 ESLint JSX 支持（ESLint 内置选项）：

```
{
  "ecmaFeatures": {
    "jsx": true
  }
}
```

然后就可以配置插件提供的规则了：

```
{
  "rules": {
    "react/display-name": 1,
    "react/jsx-boolean-value": 1
  }
}
```

3.格式化

ESLint 格式化程序

ESLint 附带有几个内置的格式化程序来控制 linting 结果的外观，并支持第三方格式化程序。 您可以在命令行上使用—format或-f标志指定格式化程序。例如，--format codeframe使用codeframe格式化程序。

- codeframe使用codeframe格式化程序。
- checkstyle
- codeframe
- compact
- html
- jslint-xml
- json
- junit
- stylish
- table
- tap
- unix
- visualstudio

以上本质是对日志输出流的格式化控制

详细查看官网： http://eslint.cn/docs/user-guide/formatters/%20frame

3.集成

集成

Eslint 常见的集成有 编辑器、构建工具、版本控制器、测试、及框架

http://eslint.cn/docs/user-guide/integrations

与webpack 集成

```
},
module: {
  rules: [
    {
      test: /\.(js|vue)$/,
      loader: 'eslint-loader',
      enforce: 'pre',
      include: [resolve('src'), resolve('test')],
      options: {
        // formatter: require('eslint-friendly-formatter'),
        // emitWarning: !config.dev.showEslintErrorsInOverlay
      }
    },
```

参考webpack 官方文档 https://webpack.js.org/loaders/eslint-loader/#src/components/Sidebar/Sidebar.jsx

与git集成

```
"scripts": {
  "dev": "webpack-dev-server --inline --progress --config build/webpack.dev.conf.js",
  "start": "npm run dev",
  "build": "BUILD_ENV=prd node build/build.js",
  "build-test": "BUILD_ENV=test node build/build.js",
  "build-pre": "BUILD_ENV=pre node build/build.js",
  "build-test-02": "BUILD_ENV=env/test node build/build.js",
  "build-pre-02": "BUILD_ENV=env/pre node build/build.js",
  "lint": "eslint --ext .js,.vue src",
  "lint-fix": "eslint --fix --ext .js,.vue src"
},
"pre-commit": [ "lint" ],
"dependencies": {
  "pre-commit": "^1.2.2",
```

首先执行 npm install pre-commit —save-dev 安装pre-commit 包，然后在pre-commit选项中配置eslint 校验的命令

原理：

```
#!/bin/zsh

function lintit () {
  OUTPUT=$(git diff --name-only | grep -E '(.js)$')
  a=("${(f)OUTPUT}")
  e=$(eslint -c eslint.json $a)
  echo $e
  if [[ "$e" != *"0 problems"* ]]; then
    echo "ERROR: Check eslint hints."
    exit 1 # reject
  fi
}
lintit
```

与前端框架 的集成 如： eslint-plugin-vue

```
module.exports = {
  extends: [
    // add more generic rulesets here, such as:
    // 'eslint:recommended',
    'plugin:vue/essential'
  ],
  rules: {
    // override/add rules settings here, such as:
    // 'vue/no-unused-vars': 'error'
  }
}
```

⚙️ Configs

This plugin provides four predefined configs:

- `plugin:vue/base` - Settings and rules to enable correct ESLint parsing
- `plugin:vue/essential` - Above, plus rules to prevent errors or unintended behavior
- `plugin:vue/strongly-recommended` - Above, plus rules to considerably improve code readability and/or dev experience
- `plugin:vue/recommended` - Above, plus rules to enforce subjective community defaults to ensure consistency

vuejs / eslint-plugin-vue

Watch 49 | Star 1,735 | Fork 161

Code | Issues 45 | Pull requests 9 | Projects 0 | Wiki | Insights

Branch: master ▾ eslint-plugin-vue / lib / rules / | Create new file | Upload files | Find file | History

michalsnik 5.0.0-beta.3 — Latest commit 3f5d41e 20 days ago

attribute-hyphenation.js	5.0.0-beta.3	20 days ago
attributes-order.js	5.0.0-beta.3	20 days ago
comment-directive.js	5.0.0-beta.3	20 days ago
component-name-in-template-casing.js	5.0.0-beta.3	20 days ago
html-closing-bracket-newline.js	5.0.0-beta.3	20 days ago
html-closing-bracket-spacing.js	5.0.0-beta.3	20 days ago
html-end-tags.js	5.0.0-beta.3	20 days ago
html-indent.js	5.0.0-beta.3	20 days ago
html-quotes.js	5.0.0-beta.3	20 days ago
html-self-closing.js	5.0.0-beta.3	20 days ago
jsx-uses-vars.js	5.0.0-beta.3	20 days ago
max-attributes-per-line.js	5.0.0-beta.3	20 days ago
multiline-html-element-content-newline.js	5.0.0-beta.3	20 days ago
mustache-interpolation-spacing.js	5.0.0-beta.3	20 days ago
name-property-casing.js	5.0.0-beta.3	20 days ago
no-async-in-computed-properties.js	5.0.0-beta.3	20 days ago
no-confusing-v-for-v-if.js	5.0.0-beta.3	20 days ago
no-dupe-keys.js	5.0.0-beta.3	20 days ago
no-duplicate-attributes.js	5.0.0-beta.3	20 days ago
no-multi-spaces.js	5.0.0-beta.3	20 days ago
no-parsing-error.js	5.0.0-beta.3	20 days ago

源码入口： https://github.com/vuejs/eslint-plugin-vue/blob/master/lib/index.js

4.进阶

规则解析原理

在分析代码前，ESlint会通过Estree将代码解析一棵抽象语法树（AST），之后ESlint会依次遍历语法树上的节点。

create方法中要返回一个object，键名对应语法树节点的类型。**ESlint在遍历语法树节点时，会执行该节点类型名所对应的回调函数。**

```
module.exports = {

    // context提供了很多实用方法，比如获取注释、获取源码...
    create: function (context) {

        return {
            CallExpression: function (node) {}
        };

    }

};
```

如`callExpression`对应的是语法树中**函数调用语句**，所以当ESlint每次遍历到函数调用语句，就会执行这句`callExpression`回调函数。

参考：https://astexplorer.net/

回调函数接受一个含有当前语句的所有信息的节点对象，我们根据这些信息来判断当前语句是否非法。通过`context.report`来抛出代码异常，传入node和message告诉ESlint代码的位置和代码错误信息：

```
module.exports = {
    create: function (context) {
        return {
            CallExpression: function (node) {

                // 如果函数名为alert，则报错
                if (node.callee.name === 'alert') {
                    context.report({
                        node: node,
                        message: 'unexpected alert'
                    });
                }
            }
        };
    }
};
```

`callee` 是 `arguments` 对象的一个属性。它可以用于引用该函数的函数体内当前正在执行的函数

AST Explorer | Snippet | JavaScript | acorn | Transform | default | ? | Parser:

```
1  /**
2   * Paste or drop some JavaScript here and explore
3   * the syntax tree created by chosen parser.
4   * You can use all the cool new features from ES6
5   * and even more. Enjoy!
6   */
7
8
9
10 var a = function (){
11     alert('123')
12 }
```

Tree | JSON

Autofocus | Hide methods | Hide empty keys | Hide location data | Hide type keys

```
start: 200
end: 217
- body: [
  - ExpressionStatement {
      type: "ExpressionStatement"
      start: 203
      end: 215
    - expression: CallExpression {
        type: "CallExpression"
        start: 203
        end: 215
      - callee: Identifier {
          type: "Identifier"
          start: 203
          end: 208
          name: "alert"
        }
      - arguments: [
        - Literal = $node {
            type: "Literal"
            start: 209
            end: 214
            value: "123"
            raw: "'123'"
          }
        ]
      }
    }
  ]
}
}
}
}
}
]
```

给 za-crm-pc 配置ESlint

总结

团队使用eslint 的难点不在于配置

而在于 定制一套能够符合大家习惯的标准，让大家去遵守，这是一个长期的过程！

参考文献：

http://eslint.cn/

https://en.wikipedia.org/wiki/JSLint

https://segmentfault.com/a/1190000010264410

https://cnodejs.org/topic/5aa2a128ce4a27f867526f7e

http://blog.lxjwlt.com/2016/09/30/eslint.html

The End

Thanks