Eslint 使用介绍及插件开发指南

Eslint 是什么

前端用来做代码检测以及错误修复的工具。

快速使用

- 安装

```
npm i -D eslint
```

- 初始化配置文件

```
./node_mosules/.bin/eslint --init
```

- 执行 eslint 命令：`eslint [options] file.js [file.js] [dir]`

```
./node_modules/.bin/eslint file.js
```

配置

配置文件

优先级：.eslintrc.js > .eslintrc.yml > .eslintrc.json > package.json(eslintConfig)

- eslint 会自动查找目录中的配置文件，不断往上层目录查找直到根目录 `/` 。如果没查找到配置文件，会使用 `~/` 目录下的配置文件。
- eslint 会将所有 .eslintrc.* 配置进行合并，对 `package.json` 的 `eslintConfig` 会进行覆盖。

> ❗ 注意：合并只是针对不同层级的配置，同个层级的不同配置文件只会使用优先级高的

利用这个特性，可以实现单个项目下不同目录有差异配置，同时享有公共配置。

```
your-project
├── .eslintrc.js
├── lib
│   └── source.js
└─┬ tests
  ├── .eslintrc.js
  └── test.js
```

评论（5）

~/

黄浩扬 2022年9月19日

8.0.0 版本后不再有这个特性

- 使用 `root: true` 可以禁止 eslint 往上查询的动作。

```
home
└── user
    └── projectA
        ├── .eslintrc.js  <- Not used
        └── lib
            ├── .eslintrc.js  <- { "root": true }
            └── main.js
```

- 使用 `-c` 配置可以指定配置文件，并只会合并上层的配置，配合 `--no-eslintrc` 可以忽略其他配置文件。（一般比较少用）

js 注释

注释的优先级高于配置文件

例如 `/* eslint-diasbled */`

配置项

rules

> eslit 内置规则

规则配置，最主要且常用的配置项。

- eslint 规则是插件化的，可以通过 rules 配置每条规则 severity，或者传递配置。
- 规则有三个等级/severity：`off/0`、`warn/1`、`error/2`
- 每条规则可以传递单独的配置，例如（一般规定第一个参数是 severity）：

```
// .eslintrc.js
module.exports = {
    rules: {
        "max-lines": [
            "warn",
            {
                max: 600, // previous 300
                skipBlankLines: true,
                skipComments: true,
            },
        ],
    }
}
```

通过注释配置规则

- 启用规则

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

// 添加配置
```

```
/* eslint eqeqeq: 0, curly: 2, quotes: ["error", "double"] */

// 添加注释
/* eslint eqeqeq: "off" -- Here's a description about why this configuration is necessary. */
```

- 禁用规则

```
// 禁用所有规则
/* eslint-disable */

alert('foo');

/* eslint-enable */


// 禁用特定规则
/* eslint-disable no-alert, no-console */

alert('foo');
console.log('bar');

/* eslint-enable no-alert, no-console */


alert('foo'); // eslint-disable-line
alert('foo'); // eslint-disable-line no-alert -- blabla

// eslint-disable-next-line
alert('foo');
// eslint-disable-next-line no-alert -- blabla
alert('foo');
```

- `noInlineConfig` 可以关闭所有注释的功能

```
// // .eslintrc.js
{
    rules: {...},
    noInlineConfig: true
}
```

extends

继承配置，可以是路径或者 npm 包。

- 递归继承，最终合并配置项。
- 路径都是参照当前配置文件位置。
- npm 包名可以省略 `eslint-config` 前缀，例如

```
// .eslintrc.js
module.exports = {
    extends: ["prettier"], // means eslint-config-prettier
}
```

> npm 包
>
> 
> 黄浩扬 2022年9月21日
> eslint-config 或 eslint-plugin 开头

- 有些 `eslint plugin` 也会导出配置用于 extends，例如

```
// .eslintrc.js
module.exports = {
    extends: ["plugin:react/recommended"], // means eslint-plugin-react/recommended
}

// eslint-plugin-react
module.exports = {
    deprecatedRules,
    rules: allRules,
    configs: {
        recommended: {...},
        all: {...},
        'jsx-runtime': {...}
    }
}
```

规则的合并

- 规则的合并分为两部分，例如

```
// Base config
"max-lines": [
    "error",
    {
        max: 300,
        skipBlankLines: true,
        skipComments: true,
    }
]


// Derived congfig 1
"max-lines": "warn"
// Resulting actual config 1
"max-lines": [
    "warn",
    {
        max: 300,
        skipBlankLines: true,
        skipComments: true,
    }
]


// Derived congfig 2
"max-lines": [
    "warn",
    {
        max: 600
    }
]
// Resulting actual config 2
"max-lines": [
    "warn",
```

```
    {
        max: 600
    }
]
```

plugins

eslint 插件，用于添加新的规则集。

- eslint 内置规则不足以提供所有开发需求，通过插件可以引入其他的规则集，引入的规则名格式为 `plugin/rule`，例如（同样，npm 包名可以省略 `eslint-plugin` 前缀）

```
// .eslintrc.js
module.exports = {
    plugins: [
        "prettier", // means eslint-plugin-prettier
        "@foobar" // means @foobar/eslint-plugin
    ],
    rules: {
        "prettier/prettier": "error", // 引入的规则名 plugin/rule
    }
}
```

- 也经常通过 plugin 来整合多个 eslint 配置 - shareable configurations，例如

```
// .eslintrc.js
module.exports = {
    extends: ["plugin:react/recommended"], // eslint-plugin-react/recommended
}
```

overrides

覆盖当前配置文件。

写法和一个完整的配置基本一样，除了没有 `root` 和 `ignorePatterns`，多了 `files` 、 `excludedFiles` 文件匹配。

```
// .eslintrc.js
module.exports = {
  "rules": {
    "quotes": ["error", "double"]
  },
  "overrides": [
    {
      "files": ["bin/*.js", "lib/*.js"],
      "excludedFiles": "*.test.js",
      "rules": {
        "quotes": ["error", "single"]
      }
    }
```

eslint-plugin-prettier

黄浩扬 2022年9月19日 （编辑过）

注意跟上文 extends 的 eslint-config-prettier 区别开

```
14      ]
15  }
```

ignorePatterns

作用同 `.eslintignore` (优先级更高)，忽略校验文件。匹配语法 glob patterns

> ignorePatterns
>
> 
> 黄浩扬 2022年9月20日
> 路径的匹配，是基于配置文件位置的。但如果是命令行，则基于工作目录。

env

eslint 提供了一组 env，每个 env 有不同的全局变量。

例如 `node` 有 `global`， `browser` 环境下有 `window`， `es6` 有 `Set` 等新特性变量。这些 env 不是互斥的，可以写多个。

```
1  // .eslintrc.js
2  module.exports = {
3      env: {
4          browser: true,
5          node: true,
6          es6: true
7      }
8  }
```

还可以通过注释指定

```
1  /* eslint-env node, browser */
2  window.temp = 'aaa'
```

globals

配置全局变量。

```
1  module.exports = {
2      globals: {
3          var1: "writable",
4          var2: "readonly"
5      }
6  }
```

parserOptions

eslint 默认解析 es5 版本代码，可以通过 parserOptions 设置 eslint 解析代码版本。

- `ecmaVersion`：指定解析代码版本，3、5(default)、6～14
- `sourceType`：指定模块类型，script(default)、module
- `ecmaFeatures`
  - `globalReturn`：允许全局 return

- `impliedStrict`：javascript strict 模式
- `jsx`：识别 jsx（但不代表识别 react 代码，需要另外引入 eslint-plugin-react）
- `project`：指定 tsconfig 配置文件

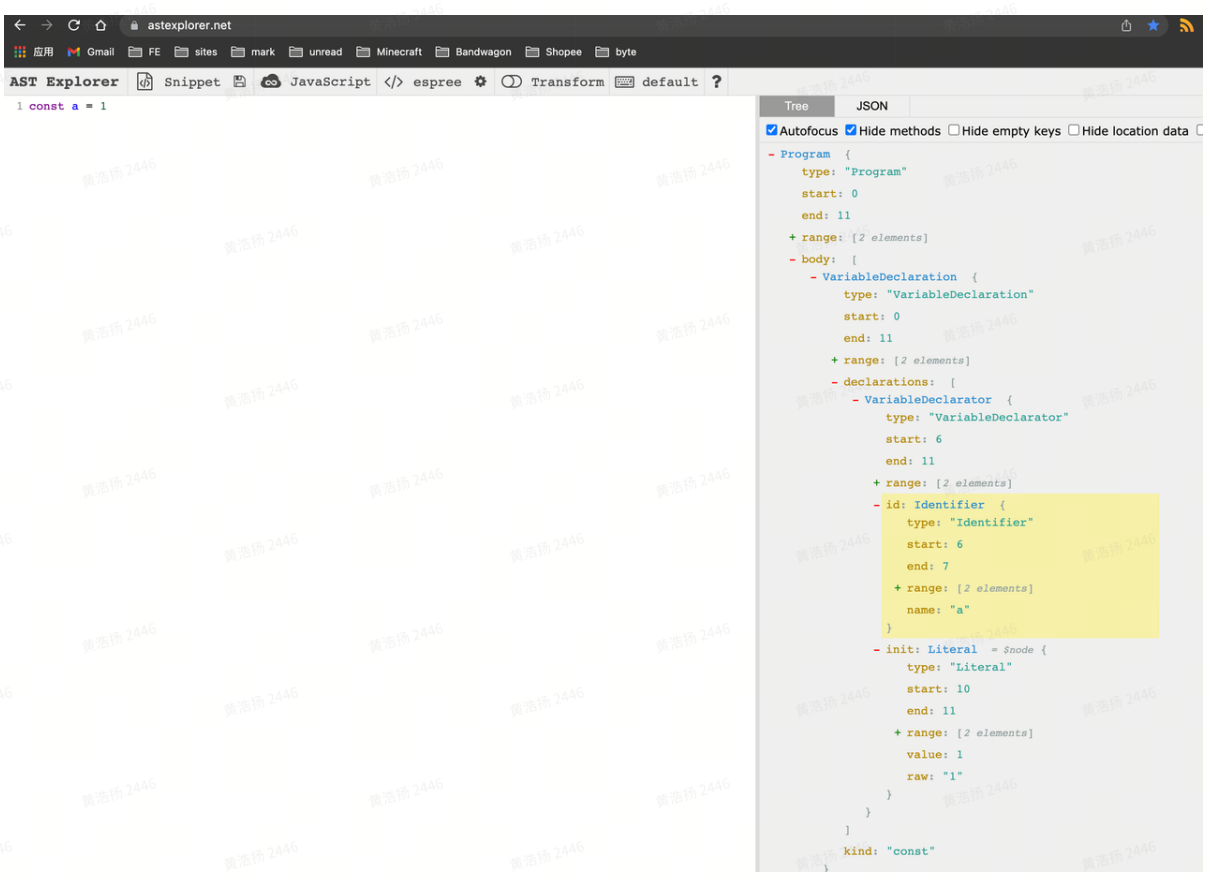
```
module.exports = {
    parserOptions: {
        ecmaVersion: 8, // es8
        sourceType: 'module',
        ecmaFeatures: {
            impliedStrict: true,
            jsx: true,
            project: "./tsconfig.js",
        }
    },
    env: {
        browser: true,
        es6: true,
    }
}
```

> ecmaVersion: 8
>
> 
> 黄浩扬 2022年9月20日
> 这里设置不代表能够解析 es8 的一些新的全局变量，全局变量需要通过 env 配置。

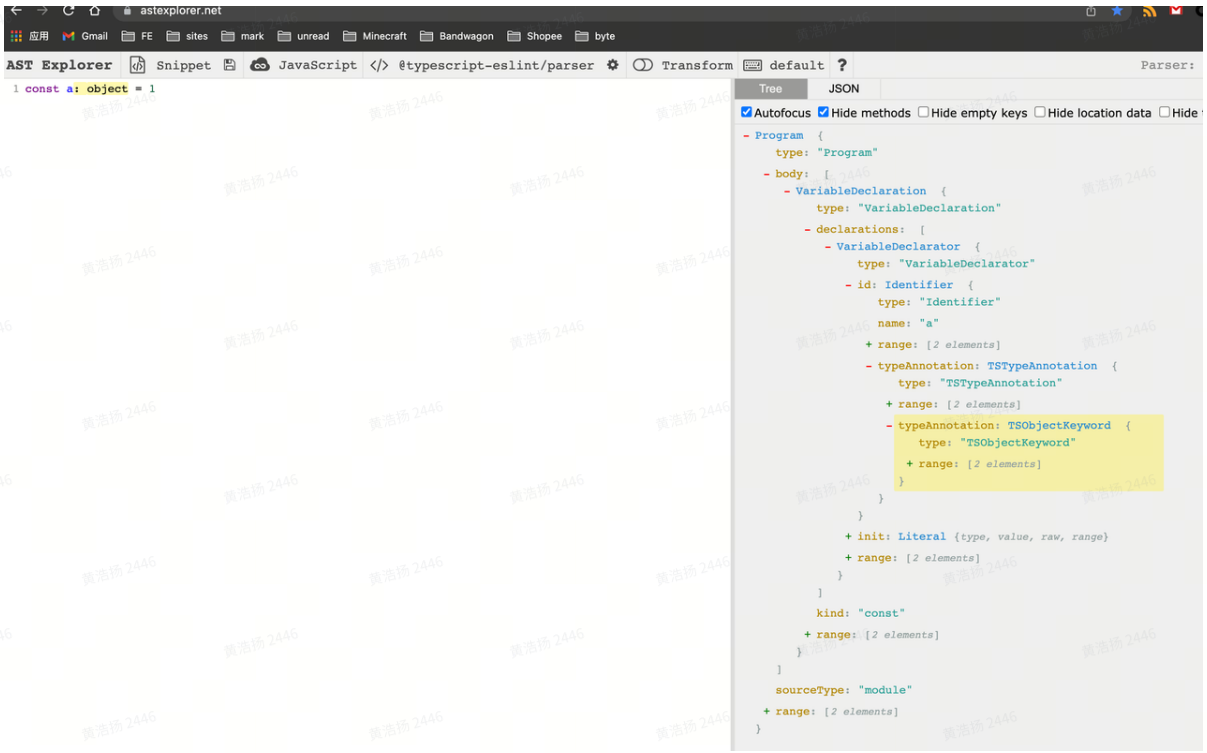
parser

> https://astexplorer.net/

- eslint 默认使用 espree 解析器，解析代码生成 AST。



- parser 可以自定义解析器。常用的解析器有：@babel/eslint-parser、@typescript-eslint/parser，它们的作用都是将代码解析成 espree 格式的 AST，同时扩展一些描述字段。



- parser 一般需要和相应的 eslint plugin 配合使用（才能使用扩展的描述字段），例如
  - @babel/eslint-parser 需要和 @babel/eslint-plugin 配合使用
  - @typescript-eslint/parser 需要和 @typescript-eslint/eslint-plugin 配合使用
- 不同的 parser 有不同的配置项，通过 parserOptions 配置。

项目中使用

项目中使用，一般会结合其他工具，实现触发式的代码检测和修复。例如配合使用 `husky` 和 `lint-staged`，实现对增量代码的 eslint 检测，并通过 git commit 钩子触发。

- 安装依赖

```
yarn add -D husky lint-staged eslint @commitlint/cli
@commitlint/config-conventional
```

- 配置 husky

```
# 安装依赖时安装 husky
npm set-script prepare "husky install"
npm run prepare

# 配置 git commit hooks，会在 commit 前触发
npx husky add .husky/pre-commit "yarn lint"
```

- package.json 中配置 lint-staged

```
// package.json
{
    "scripts": {
        "lint": "lint-staged"
    }
    "lint-staged": {
```

```
        "*.{ts,tsx,js,jsx}": "eslint --fix --color --quiet --
    cache",
        "git add"
    }
}
```

- 添加 commit-lint 规范 commit 信息

```
// package.json
{
    "commitlint": {
        "extends": [
          "@commitlint/config-conventional"
        ]
    }
}
```

- 安装 vscode eslint 插件，保存代码时自动格式化（eslint --fix）

```
// vscode setting。json
{
    "editor.formatOnSave": true,
    "editor.codeActionsOnSave": {
        "source.fixAll.eslint": true
    },
}
```

开发一个 eslint 插件

脚手架

- 安装依赖

```
npm i -g yo generator-eslint
```

- 通过 yo 脚手架生成 plugin 目录

```
 yo eslint:plugin
```

```
⚙️ bytedance@C02GF2C2MD6R  ~/Desktop/bytedance/pro/eslint-
plugin  yo eslint:plugin

? What is your name? @s-fe/eslint-plugin
? What is the plugin ID? s-fe
? Type a short description of this plugin: eslint guidelines
for S-FE
? Does this plugin contain custom ESLint rules? Yes
? Does this plugin contain one or more processors? No
```

```
    create package.json
    create .eslintrc.js
    create lib/index.js
    create README.md
```

- 通过脚手架生成 rule 目录

```
yo eslint:rule
```

```
⚙ bytedance@C02GF2C2MD6R  ~/Desktop/bytedance/pro/eslint-plugin  dev  yo eslint:rule
? What is your name? test-rule
? Where will this rule be published? ESLint Plugin
? What is the rule ID? test-rule
? Type a short description of this rule: the second param of setTimeout is forbidden to use number
? Type a short example of the code that will fail: setTimeout(() => {}, 2)
   create docs/rules/test-rule.md
   create lib/rules/test-rule.js
   create tests/lib/rules/test-rule.js
```

- 目录结构

```
⚙ bytedance@C02GF2C2MD6R  ~/Desktop/bytedance/pro/eslint-plugin  dev  tree -I node_modules
.
├── README.md
├── docs
│   └── rules
│       └── test-rule.md
├── lib
│   ├── index.js
│   └── rules
│       └── test-rule.js
├── package-lock.json
├── package.json
└── tests
    └── lib
        └── rules
            └── test-rule.js
```

编写插件

插件必须输出一个 `rules` 对象，包含规则 ID 和对应规则的一个键值对。

编写规则

> 官方编写规则文档

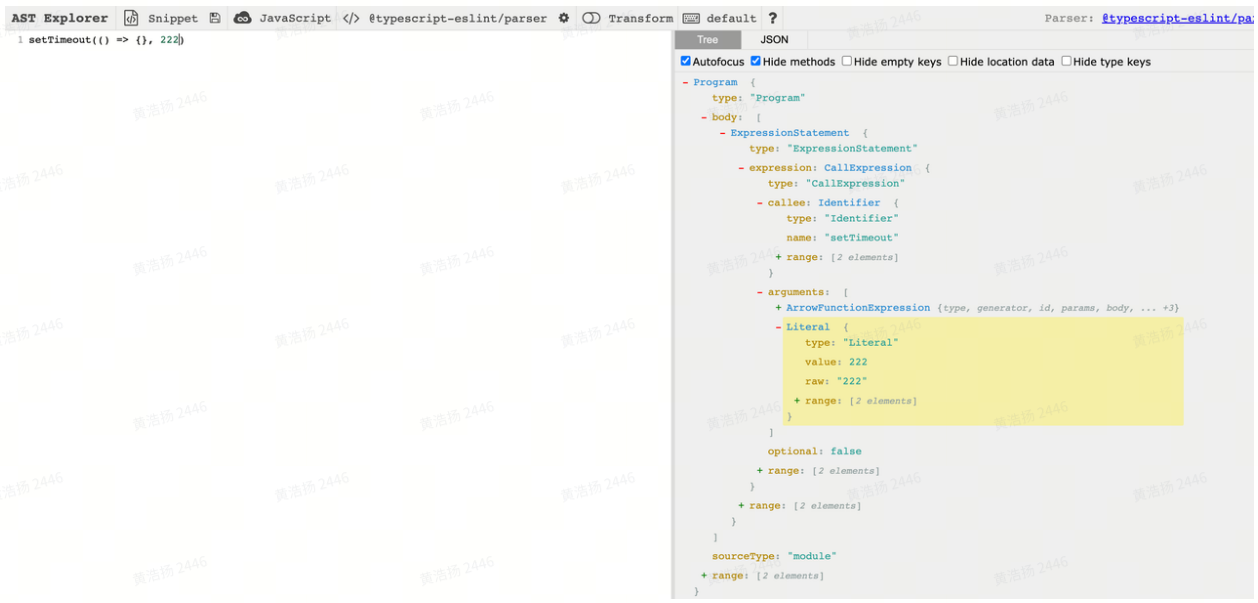
生成的规则模版如下，主要包括两部分：

- meta：规则描述，以及 eslint 修复配置
- create：规则实现函数，返回一个对象，对象 key 为 eslint 暴露的钩子，value 为回调函数
  - key 是 Selector，主要是 espree AST 的一个节点类型，在 `向下` 遍历 AST 时，ESLint 调用回调函数
  - key 的基础上加了 `:exit`，则在 `向上` 遍历 AST 时，ESLint 调用回调函数
  - key 是个事件名字，eslint 为代码路径分析调用 handler 函数

```
/**
 * @fileoverview the second param of setTimeout is forbidden to use number
 * @author test-rule
 */
"use strict";

//------------------------------------------------------------------------------
// Rule Definition
//------------------------------------------------------------------------------

/** @type {import('eslint').Rule.RuleModule} */
module.exports = {
  meta: {
    type: 'suggestion', // `problem`, `suggestion`, or `layout`
    docs: {
      description: "the second param of setTimeout is forbidden to use number",
      recommended: false,
      url: null, // URL to the documentation page for this rule
    },
    fixable: null, // Or `code` or `whitespace`
    schema: [], // Add a schema if the rule has options
  },

  create(context) {
    // variables should be defined here

    //----------------------------------------------------------------------
    // Helpers
    //----------------------------------------------------------------------

    // any helper functions should go here or else delete this section

    //----------------------------------------------------------------------
    // Public
    //----------------------------------------------------------------------

    return {
      // visitor functions for different types of nodes
    };
  },
};
```

```
42
```

参考 AST 编写规则




```js
/**
 * @fileoverview the second param of setTimeout is forbidden to use number
 * @author test-rule
 */
"use strict";

//------------------------------------------------------------------------------
// Rule Definition
//------------------------------------------------------------------------------

/** @type {import('eslint').Rule.RuleModule} */
module.exports = {
  meta: {
    type: 'problem', // `problem`, `suggestion`, or `layout`
    docs: {
      // description: "the second param of setTimeout is forbidden to use number",
      // recommended: false,
      url: null, // URL to the documentation page for this rule
    },
    fixable: null, // Or `code` or `whitespace`
    schema: [], // Add a schema if the rule has options
    messages: {
      messageId: 'setTimeout 第二个参数禁止是数字',
    },
  },

  create(context) {
    // variables should be defined here

    //----------------------------------------------------------------------
    // Helpers
    //----------------------------------------------------------------------

```

```
    // any helper functions should go here or else delete this section

    //--------------------------------------------------------------
    // Public
    //--------------------------------------------------------------

    return {
      // visitor functions for different types of nodes
      'CallExpression': (node) => {
        if (node.callee.name !== 'setTimeout') return // 不是 setTimeout 直接过滤

        const timeNode = node.arguments && node.arguments[1] // 获取第二个参数
        if (!timeNode) return

        if (timeNode.type === 'Literal' && typeof timeNode.value === 'number') {
          context.report({
              node,
              // message: ''
              messageId: 'messageId'
          })
        }
      }
    };
  },
};

```

添加自动修复功能

- fixable: 'code' 开启修复功能
- context.report() 提供一个 fix 函数

```
module.exports = {
    meta: {
        type: 'suggestion', // `problem`, `suggestion`, or `layout`
        docs: {
          description: "the second param of setTimeout is forbidden to use number",
          recommended: false,
          url: null, // URL to the documentation page for this rule
        },
        fixable: 'code', // Or `code` or `whitespace`
        schema: [], // Add a schema if the rule has options
        messages: {
          messageId: 'setTimeout 第二个参数禁止是数字',
        },
    },
    create: function(context) {
      return {
```

```
        'CallExpression': (node) => {
          if (node.callee.name !== 'setTimeout') return // 不是 setTimeout 直接过滤

          const timeNode = node.arguments && node.arguments[1] // 获取第二个参数
          if (!timeNode) return

          if (timeNode.type === 'Literal'  && typeof timeNode.value === 'number') {
            context.report({
              node,
              message: 'setTimeout 第二个参数禁止是数字',
              fix(fixer) {
                const numberValue = timeNode.value;
                const statementString = `var timeoutNum = ${numberValue}\n`;
                return [
                  fixer.replaceTextRange(node.arguments[1].range, 'timeoutNum'),
                  fixer.insertTextBeforeRange(node.range, statementString)
                ]
              }
            })
          }
        }
      };
    }
}
```

编写单测

验证规则的有效性

```
/**
 * @fileoverview the second param of setTimeout is forbidden to use number
 * @author test-rule
 */
"use strict";

//------------------------------------------------------------------------------
// Requirements
//------------------------------------------------------------------------------

const rule = require("../../../lib/rules/test-rule"),
  RuleTester = require("eslint").RuleTester;

//------------------------------------------------------------------------------
// Tests
//------------------------------------------------------------------------------

const ruleTester = new RuleTester();
```

```
ruleTester.run("test-rule", rule, {
  valid: [
    // give me some code that won't trigger a warning
    {
      code: `
        var num = 1000;
        setTimeout(function() { console.log(2) }, num)
      `,
    },
  ],

  invalid: [
    {
      code: "setTimeout(function() {}, 2)",
      errors: [{ messageId: "messageId", type: "CallExpression" }],
      output: 'var timeoutNum = 2;\nsetTimeout(function() {}, timeoutNum)'
    },
  ],
});

```

项目中使用

```
// .eslintrc.js
{
    plugins: ["@s-fe"], // @s-fe/eslint-plugin
    rules: {
        "@s-fe/test-rule": "error"
    }
}
```

整合规则集

因为 plugin 的规则默认是不启用的，如果导出规则太多，可以通过 plugins 导出规则集进行 extends。

```
module.exports = {
  rules:  requireIndex(__dirname + "/rules"),
  configs: {
    recommended: {
      plugins: ['@s-fe'],
      rules: {
        '@s-fe/test-rule': 'error'
      }
    }
  }
}
```

使用

```
// .eslintrc.js
{
    extends: ["plugin:@s-fe/recommended"]
}
```

> https://juejin.cn/post/7025256331120476197
>
> 自定义 Eslint 开发