简介

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，这样我们就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**。

ESLint 由 JavaScript 红宝书 作者 Nicholas C. Zakas 编写， 2013 年发布第一个版本。 NCZ 的初衷不是重复造一个轮子，而是在实际需求得不到 JSHint 团队响应 的情况下做出的选择：以可扩展、每条规则独立、不内置编码风格为理念编写一个 lint 工具。

ESLint 主要有以下特点：

- 默认规则包含所有 JSLint、JSHint 中存在的规则，易迁移；
- 规则可配置性高：可设置「警告」、「错误」两个 error 等级，或者直接禁用；
- 包含代码风格检测的规则（可以丢掉 JSCS 了）；
- 支持插件扩展、自定义规则。

安装和使用

先决条件：Node.js (>=4.x), npm version 2+。
有两种方式安装 ESLint： 全局安装和本地安装。

如果你想让 ESLint 成为你项目构建系统的一部分，我们建议在本地安装。你可以使用 npm：

```
$ npm install eslint --save-dev
```

紧接着你应该设置一个配置文件：

```
$ ./node_modules/.bin/eslint --init
```

之后，你可以在你项目根目录运行 ESLint：

```
$ ./node_modules/.bin/eslint yourfile.js
```

使用本地安装的 ESLint 时，你使用的任何插件或可分享的配置也都必须在本地安装。

配置

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。

1） .eslintrc 文件示例：

```
// 环境定义
"env": {
  "browser": true,
},
//  JavaScript 语言选项
"parserOptions": {
  "ecmaVersion": 6,
  "ecmaFeatures": {
    "jsx": true
  }
},
// 全局变量
"globals": {
  "angular": true,
},
// 规则
"rules": {
  "camelcase": 2,
  "curly": 2,
  "brace-style": [2, "1tbs"],
  "quotes": [2, "single"],
  "semi": [2, "always"],
  "space-in-brackets": [2, "never"],
  "space-infix-ops": 2,
 },
// 使用第三方插件
"plugins": [
  "plugin1",
```

```
    "eslint-plugin-plugin2"
  ]
  //
  extends: 'eslint:recommended',
}
```

.eslintrc 放在项目根目录，则会应用到整个项目；如果子目录中也包含 .eslintrc 文件，则子目录会忽略根目录的配置文件，应用该目录中的配置文件。这样可以方便地对不同环境的代码应用不同的规则。

插件，继承

ESLint 支持使用第三方插件。在使用插件之前，你必须使用 npm 安装它。

在配置文件里配置插件时，可以使用 plugins 关键字来存放插件名字的列表。插件名称可以省略 eslint-plugin- 前缀。可在npm官网查询所需的插件

继承是引用其他eslint配置文件规则

2） package.json 示例：

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

3） 文件内配置

代码文件内配置的规则会覆盖配置文件里的规则。

禁用 ESLint：

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

禁用一条规则：

```
/*eslint-disable no-alert */
alert('doing awful things');
/* eslint-enable no-alert */
```

调整规则：

```
/* eslint no-comma-dangle:1 */
// Make this just a warning, not an error
var obj = { key: 'value',
```

忽略检查

你可以通过在项目根目录创建一个 .eslintignore 文件告诉 ESLint 去忽略特定的文件和目录。.eslintignore 文件是一个纯文本文件，其中的每一行都是一个 glob 模式表明哪些路径应该忽略检测。

例如：把下面 .eslintignore 文件放到当前工作目录里，将忽略项目根目录下的 node_modules，bower_components 以及 build/ 目录下除了 build/index.js 的所有文件。

```
# /node_modules/* and /bower_components/* in the project root are ignored by
default

# Ignore built files except build/index.js
build/*
!build/index.js
```

重要：注意代码库的 node_modules 目录，比如，一个 packages 目录，默认情况下不会被忽略，需要手动添加到 .eslintignore。

常用命令

禁用 .eslintrc.* 和 package.json 文件中的配置。

```
eslint --no-eslintrc file.js
```

-c, --config
该选项允许你为 ESLint (查看 Configuring ESLint 了解更多)指定一个额外的配置文件。

```
eslint -c ~/my-eslint.json file.js
```

工作流集成

ESLint 可以集成到主流的编辑器和构建工具中，以便我们在编写的代码的同时进行 lint。

编辑器集成

以 WebStorm 为例，只要全局安装 ESLint 或者在项目中依赖中添加 ESLint ，然后在设置里开启 ESLint 即可。其他编辑可以从官方文档中获得获得具体信息。

构建系统集成

在 Gulp 中使用：

```
var gulp = require('gulp');
var eslint = require('gulp-eslint');

gulp.task('lint', function() {
  return gulp.src('client/app/**/*.js')
    .pipe(eslint())
    .pipe(eslint.format());
});
```

代码风格检测

在团队协作中，统一的代码风格更具可读性、可维护性。ESLint 内置了一系列有关代码风格的规则，可以根据团队的编码规范设置。

自定义规则

显然，ESLint 内置的规则不可能包罗所有需求。可以通过插件实现自定义规则，这是 ESLint 最有卖点的功能。在 NPM 上以 eslintplugin 为关键词，可以搜索到很多插件，包括 eslint-plugin-react。如果有自行开发插件的需求，可以阅读 ESLint 插件开发文档。

延深思考：
ESLint、jshint于TSLint三者是什么关系

官网链接参考：

- 配置
- 命令行
- 规则
- 集成