第十三节 ESLint

1.定义

- 最为主流的JavaScript Lint工具检测JS代码质量
- ESLint很容易统一开发者的编码风格
  - 缩进
  - 代码优化建议
  - 禁止使用那些编码习惯,例如三等和双等
- ESLint可以帮助开发者提升编码能力

2.ESLint的安装

- 初始化项目

```
yarn init -y
```

- 安装ESLint模块为开发依赖

```
yarn add eslint -D
```

- 通过CLI命令验证安装结果

```
yarn eslint --version // 查看版本
```

3.ESLint快速上手

- 编写问题代码

```
const foo=123

function fn() {
    console.log("hello");
        console.log("eslint");
}

fn(

syy()
```

- 使用eslint配置文件

```
yarn eslint --init //初始化eslint配置文件
```

```
→ 04-01-eslint yarn eslint --init
yarn run v1.22.4
$ E:\workspace_learn\fed-e-code\part-02\module-02\04-01-eslint\node_modules\.bin\eslint --init
? How would you like to use ESLint?
  To check syntax only
> To check syntax and find problems
  To check syntax, find problems, and enforce code style
```

1. 只检测语法错误
2. 检测语法错误,并发现不合理的地方(例如,未使用的变量)
3. 检测语法错误,发现不合理的地方, 检测代码风格

```
? What type of modules does your project use? (Use arrow keys)
> JavaScript modules (import/export)
  CommonJS (require/exports)
  None of these
```

使用哪个模块化标准,(其实是允许你使用那些语法啊,API等)

1. ES Module
2. CommmonJS
3. 无

```
? Which framework does your project use? (Use arrow keys)
> React
  Vue.js
  None of these
```

使用那个前端框架

1. React
2. Vue.js
3. 无

```
? Does your project use TypeScript? (y/N)
```

是否使用typescript

```
? Where does your code run? (Press <space> to select, <a> to toggle all, <i> to invert selection)
>(*) Browser
 ( ) Node
```

代码运行在哪个环境

1. 浏览器
2. node后端程序

```
? How would you like to define a style for your project? (Use arrow key
> Use a popular style guide
  Answer questions about your style
  Inspect your JavaScript file(s)
```

你想怎么定义的项目代码风格

1. 使用流行的风格
2. 根据询问的问题来选择
3. 根据js文件来配置

```
? Which style guide do you want to follow? (Use arrow keys)
> Airbnb: https://github.com/airbnb/javascript
  Standard: https://github.com/standard/standard
  Google: https://github.com/google/eslint-config-google
```

选择哪个流行的代码风格,一般选择standard

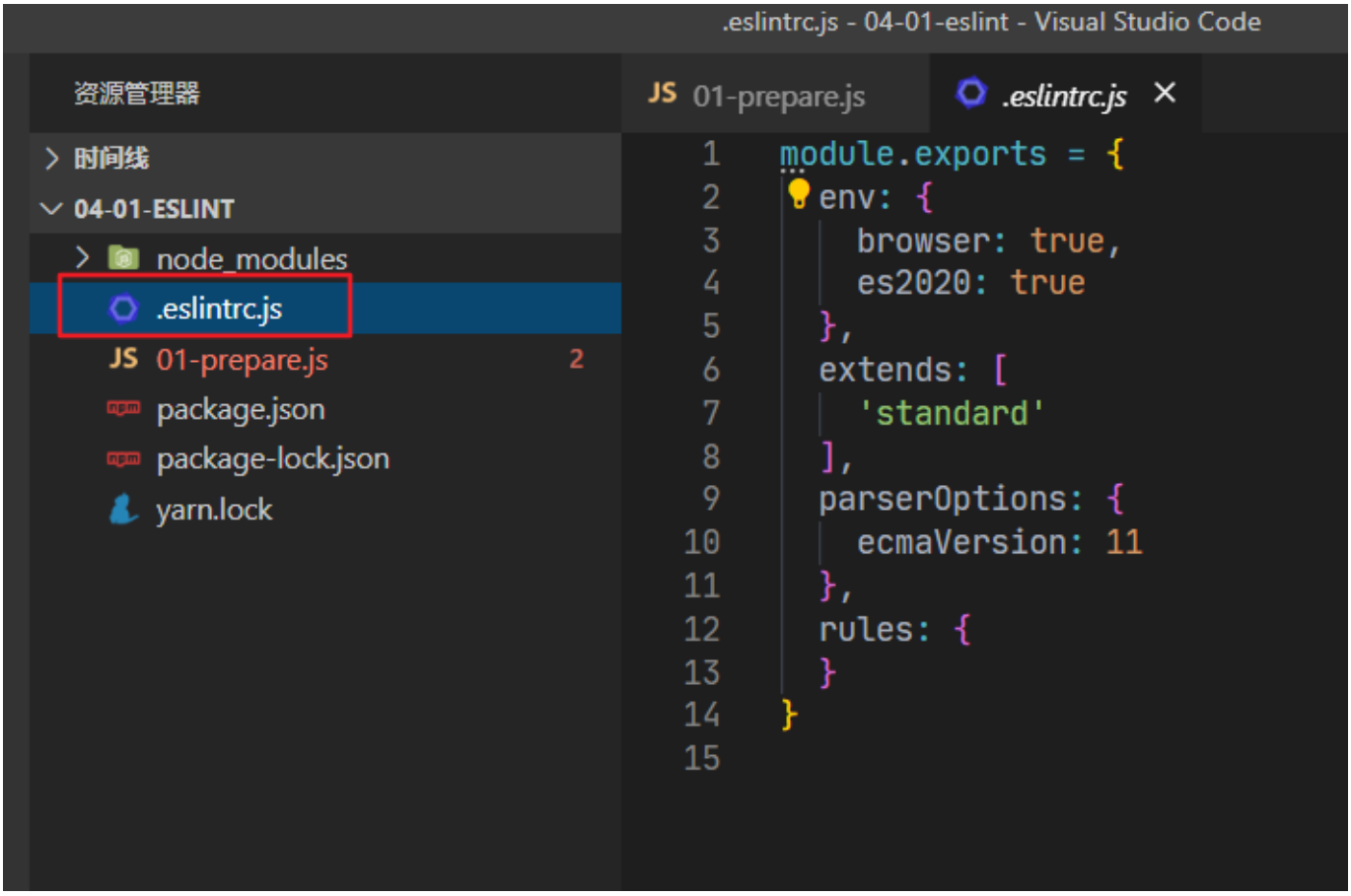
```
? What format do you want your config file to be in? (Use arrow keys)
> JavaScript
  YAML
  JSON
```

配置文件想用那种文件格式书写

```
Checking peerDependencies of eslint-config-standard@latest
The config that you've selected requires the following dependencies:

eslint-config-standard@latest eslint@>=6.2.2 eslint-plugin-import@>=2.18.0 eslint-plugin-node@>=9.1.0
eslint-plugin-promise@>=4.2.1 eslint-plugin-standard@>=4.0.0
? Would you like to install them now with npm? (Y/n)
```

根据刚刚选择的风格安装插件

生成了配置文件



- 使用eslint`执行检测

```
yarn eslint 01-prepare.js
```

检查到语法错误之后,eslint是没法检查代码问题和语法风格的所以就是这么少

```
→ 04-01-eslint yarn eslint 01-prepare.js
yarn run v1.22.4
$ E:\workspace_learn\fed-e-code\part-02\module-02\04-01-eslint\node_mod
ules\.bin\eslint 01-prepare.js

E:\workspace_learn\fed-e-code\part-02\module-02\04-01-eslint\01-prepare
.js
  10:6  error  Parsing error: Unexpected token

✖ 1 problem (1 error, 0 warnings)

error Command failed with exit code 1.
info Visit https://yarnpkg.com/en/docs/cli/run for documentation about
this command.
```

修改代码之后

```
1  const foo=123
2
3  function fn() {
4      console.log("hello");
5          console.log("eslint");
6  }
7
8  fn()
9
10 syy()
```

```
E:\workspace_learn\fed-e-code\part-02\module-02\04-01-eslint\01-prepare
.js
   1:7   error  'foo' is assigned a value but never used       no-unuse
d-vars
   1:10  error  Operator '=' must be spaced                    space-in
fix-ops
   3:12  error  Missing space before function parentheses      space-be
fore-function-paren
   4:1   error  Expected indentation of 2 spaces but found 4   indent
   4:17  error  Strings must use singlequote                   quotes
   4:25  error  Extra semicolon                                semi
   5:1   error  Expected indentation of 2 spaces but found 8   indent
   5:21  error  Strings must use singlequote                   quotes
   5:30  error  Extra semicolon                                semi
  10:1   error  'syy' is not defined                           no-undef
  10:6   error  Newline required at end of file but not found  eol-last

✖ 11 problems (11 errors, 0 warnings)
  9 errors and 0 warnings potentially fixable with the `--fix` option.

error Command failed with exit code 1.
info Visit https://yarnpkg.com/en/docs/cli/run for documentation about
this command.
```

使用 --fix修复大部分的代码风格问题

```
yarn eslint 01-prepare.js  --fix
```

```
1  const foo = 123
2
3  function fn () {
4    console.log('hello')
5    console.log('eslint')
6  }
7
8  fn()
9
10 syy()
11
```

4.ESLint配置文件解析

配置文件会影响当前目录和子目录的所有文件

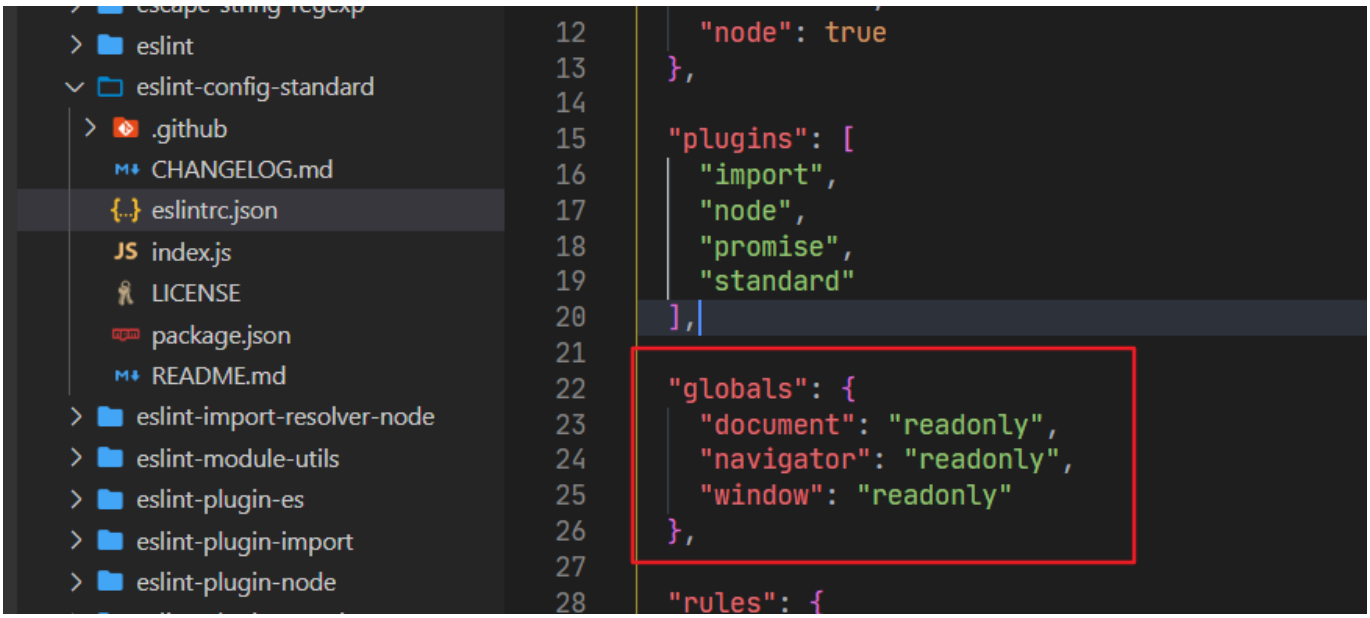
4.1 env 标识当前项目的运行环境

如果把env中的browser设置为false,
代码中使用了document的api,但是没有报错
原因是我们的代码风格继承了standard,他全局对象中包含了golbals
如果使用了alert就会报错

```
document.createElement('ul')

alert('12')
```

```
module.exports = {
  env: {
    browser: false, // 设置为false
    es2020: true
  },
  extends: [
    'standard'
  ],
  parserOptions: {
    ecmaVersion: 11
  },
  rules: {
  }
}
```

```
E:\workspace_learn\fed-e-code\part-02\module-02\04-01-eslint\02-configu
ration.js
  3:1  error  'alert' is not defined  no-undef

✖ 1 problem (1 error, 0 warnings)

error Command failed with exit code 1.
info Visit https://yarnpkg.com/en/docs/cli/run for documentation about
this command.
```



env所支持的运行环境,可以同时开启多个运行环境

4.2 extends使用了那个风格进行继承

4.3 parserOptions

- exmaVersion 使用es那个版本的语法 进行检测,并不代表不能使用那些语法

4.4 rules 代码风格检测

- `"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）

```
{
    "rules": {
```

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

4.5 globals

可以使用那些全局变量

```
module.exports = {
  env: {
    browser: false, // 设置为false
    es2020: true
  },
  extends: [
    'standard'
  ],
  parserOptions: {
    ecmaVersion: 11
  },
  rules: {
  },
  globals: {
    jQuery: 'readonly'
  }
}
```

5. eslint 配置注释

5.1可以在你的文件中使用以下格式的块注释来临时禁止规则出现警告

```
/* eslint-disable */

alert('foo');

/* eslint-enable */
```

5.2 如果在整个文件范围内禁止规则出现警告，将 `/* eslint-disable */` 块注释放在文件顶部

```
/* eslint-disable */

alert('foo');
```

5.3 你也可以对整个文件启用或禁用某个规则:

```
/* eslint-disable no-alert */


alert('foo');
```

5.3 在某一特定的行上禁用所有规则：

```
alert('foo'); // eslint-disable-line

// eslint-disable-next-line
alert('foo');

/* eslint-disable-next-line */
alert('foo');

// eslint-disable-next-line no-alert 禁用某个规则
alert('foo');

alert('foo'); /* eslint-disable-line no-alert, quotes, semi */ // 禁用多个规则
```

5.4 禁用某个插件的风格

禁止 `eslint-plugin-example` 的 `rule-name` 规则，把插件名（`example`）和规则名（`rule-name`）结合为 `example/rule-name`：

```
1 foo(); // eslint-disable-line example/rule-name
```

6.ESLint结合gulp使用

1.安装eslint / gulp-eslint

```
1 yarn add eslint gulp eslint -D
```

2.创建eslint配置文件

```
1 yarn eslint --init
```

3.修改gulpfile.js

```
1  const script = () => {
2      return src('src/assets/scripts/*.js', { base: 'src' })
3          .pipe(plugins.eslint.format()) // --fixed
4          .pipe(plugins.eslint.failAfterError()) // 发现错误终止编译并且打
   印到控制台
5          .pipe(
6              plugins.babel({
7                  presets: ['@babel/preset-env'],
8              })
9          )
10         .pipe(dest('temp'));
11 };
```

7.ESLint结合webpack使用

1.安装eslint /eslint-loader

```
yarn add eslint eslint-loader
```

2.添加eslint配置文件

```
yarn eslint --init
```

3.webpack中使用eslint-loader

```
const HtmlWebpackPlugin = require('html-webpack-plugin')

module.exports = {
  mode: 'production',
  entry: './src/main.js',
  module: {
    rules: [
      {
        test: /\.js$/,
        exclude: /node_modules/,
        use: 'babel-loader'
      },
      {
        test: /\.js$/,
        exclude: /node_modules/,
        use: 'eslint-loader',
        enforce: 'pre' // 优先使用该loader
      },
      {
        test: /\.css$/,
        use: [
          'style-loader',
          'css-loader'
        ]
      }
    ]
  },
  plugins: [
    new HtmlWebpackPlugin({
      template: 'src/index.html'
```

```
    })
  ]
}
```

针对react项目做的一个插件
目的是增强代码的检查,并对react的语法做成优化建议

```
yarn add eslint-plugin-react -D
```

.eslintrc.js添加插件

```
module.exports = {
  env: {
    browser: true,
    es2020: true
  },
  extends: [
    'plugin:react/recommended', // 继承插件中的规则,也可以在rules中单独使用,看情况
    'standard'
  ],
  parserOptions: {
    ecmaFeatures: {
      jsx: true
    },
    ecmaVersion: 11,
    sourceType: 'module'
  },
  plugins: [
    'react' // 使用插件
  ],
  rules: {
  }
}
```

8.ESLint检查TypeScript

1.初始化eslint配置文件

```
yarn eslint --init
```

初始化的过程中添加对tyescript的支持
也可以在现有的项目中添加插件

```
yarn add @typescript-eslint/parser -D
```

然后添加到.eslintrc.js文件中

```
module.exports = {
  env: {
    browser: true,
    es2020: true
  },
  extends: [
    'plugin:react/recommended',
    'standard'
  ],
  parser: '@typescript-eslint/parser',
  parserOptions: {
    ecmaFeatures: {
      jsx: true
    },
    ecmaVersion: 11,
    sourceType: 'module'
  },
  plugins: [
    'react'
  ],
  rules: {
  }
}
```