# 前端工程化

## —ESLint基础

# 01 ESLint介绍及特性

ESLint是一个用来识别 ECMAScript 并且按照规则给出报告的代码检测工具，使用它可以避免低级错误和统一代码的风格。如果每次在代码提交之前都进行一次eslint代码检查，就不会因为某个字段未定义为undefined或null这样的错误而导致服务崩溃，可以有效的控制项目代码的质量。在许多方面，ESLint类似于 JSLint 和 JSHint，但有一些例外：

1. ESLint 使用 Espree 进行 JavaScript 解析。
2. ESLint 使用 AST 来评估代码中的模式。
3. ESLint 是完全可插拔的，每一条规则都是一个插件，可以在运行时添加更多。

中文文档：http://nodejs.cn/eslint/

# 01 ESLint介绍及特性

1. 发现问题： ESLint静态分析代码以快速发现问题。ESLint内置于大多数文本编辑器中，可以将ESLint作为持续集成管道的一部分运行。
2. 自动修复： ESLint发现的许多问题可以自动修复。ESLint修复程序具有语法意识，因此不会遇到传统查找和替换算法引入的错误。
3. 定制：预处理代码，使用自定义解析器，并编写与ESLint内置规则一起使用的自己的规则。可以自定义ESLint，使其完全按照项目所需的方式工作。

出处：https://juejin.cn/post/7007419705543622669#heading-20

# 02 ESLint解决什么问题

```
var hasBarProperty = foo.hasOwnProperty("bar");
var isPrototypeOfBar = foo.isPrototypeOf(bar);
var barIsEnumerable = foo.propertyIsEnumerable("bar");
```

# 02 ESLint解决什么问题

```
1  var hasBarProperty = Object.prototype.hasOwnProperty.call(foo, "bar");
2  var isPrototypeOfBar = Object.prototype.isPrototypeOf.call(foo, bar);
3  var barIsEnumerable = {}.propertyIsEnumerable.call(foo, "bar");
4
```

# 02 ESLint解决什么问题

```
var hasBarProperty = foo.hasOwnProperty("bar");
var isPrototypeOfBar = foo.isPrototypeOf(bar);
var barIsEnumerable = foo.propertyIsEnumerable("bar");
```

不好

```
1  var hasBarProperty = Object.prototype.hasOwnProperty.call(foo, "bar");
2  var isPrototypeOfBar = Object.prototype.isPrototypeOf.call(foo, bar);
3  var barIsEnumerable = {}.propertyIsEnumerable.call(foo, "bar");
4
```

好

# 03 工作原理

分为两个步骤:

1、使用 Espree 解析 JavaScript

2、使用 AST 去分析代码中的模式

https://astexplorer.net/

# 03 工作原理

var hasBarProperty =

Object.prototype.hasOwnProperty.call(foo, "bar");

# 04 使用ESLint

3个步骤：

1、安装和初始化

2、配置规则

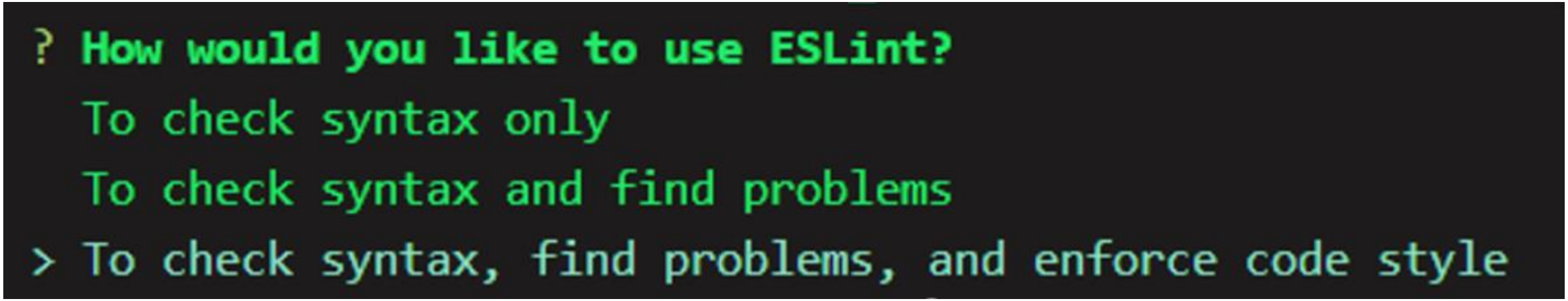
3、检验代码

ESLint需要在node环境下运行，安装ESLint前先安装Node.js环境。

# 04 使用ESLint

安装： npm install eslint --save-dev

调用eslint： ./node_modules/eslint/bin/eslint.js –init



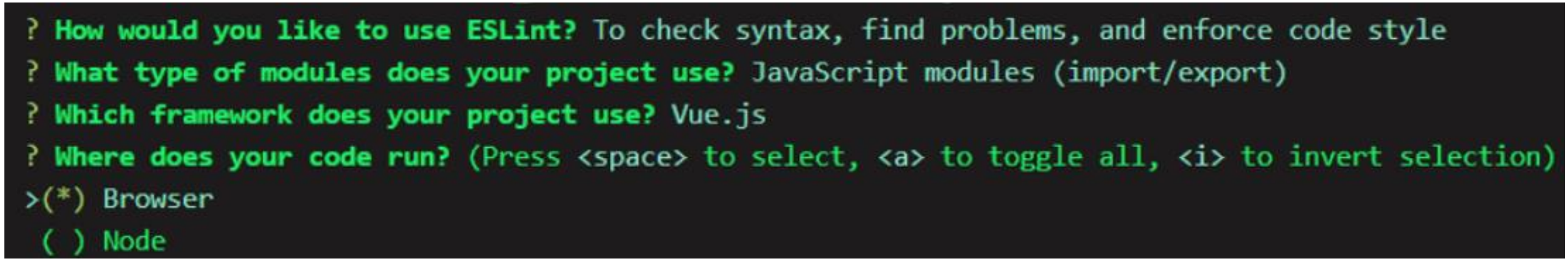
# 04 使用ESLint

```
? How would you like to use ESLint? To check syntax, find problems, and enforce code style
? What type of modules does your project use? (Use arrow keys)
> JavaScript modules (import/export)
  CommonJS (require/exports)
  None of these
```

# 04 使用ESLint

```
? How would you like to use ESLint? To check syntax, find problems, and enforce code style
? What type of modules does your project use? JavaScript modules (import/export)
? Which framework does your project use?
  React
> Vue.js
  None of these
```

# 04 使用ESLint


? How would you like to use ESLint? To check syntax, find problems, and enforce code style
? What type of modules does your project use? JavaScript modules (import/export)
? Which framework does your project use? Vue.js
? Where does your code run? (Press <space> to select, <a> to toggle all, <i> to invert selection)
>(*) Browser
( ) Node

# 04 使用ESLint

```
? How would you like to use ESLint? To check syntax, find problems, and enforce code style
? What type of modules does your project use? JavaScript modules (import/export)
? Which framework does your project use? Vue.js
? Where does your code run? Browser
? How would you like to define a style for your project? (Use arrow keys)
> Use a popular style guide
  Answer questions about your style
  Inspect your JavaScript file(s)
```

# 04 使用ESLint

```
? How would you like to use ESLint? To check syntax, find problems, and enforce code style
? What type of modules does your project use? JavaScript modules (import/export)
? Which framework does your project use? Vue.js
? Where does your code run? Browser
? How would you like to define a style for your project? Use a popular style guide
? Which style guide do you want to follow?
  Airbnb (https://github.com/airbnb/javascript)
> Standard (https://github.com/standard/standard)
  Google (https://github.com/google/eslint-config-google)
```

# 04 使用ESLint

```
? How would you like to use ESLint? To check syntax, find problems, and enforce code style
? What type of modules does your project use? JavaScript modules (import/export)
? Which framework does your project use? Vue.js
? Where does your code run? Browser
? How would you like to define a style for your project? Use a popular style guide
? Which style guide do you want to follow? Standard (https://github.com/standard/standard)
? What format do you want your config file to be in? (Use arrow keys)
> JavaScript
  YAML
  JSON
```

# 04 使用ESLint

经过上述步骤配置完毕后，项目会生成文件.eslintrc.js

```
module.exports = {
  env: {
    browser: true,
    es2021: true
  },
  extends: [
    'plugin:vue/vue3-essential',
    'standard'
  ],
  overrides: [
  ],
  parserOptions: {
    ecmaVersion: 'latest',
    sourceType: 'module'
  },
  plugins: [
    'vue'
  ],
  rules: {
  }
}
```

# 04 使用ESLint

经过上述步骤配置完毕后，项目会生成文件.eslintrc.js

```
module.exports = {
  env: {
    browser: true,
    es2021: true
  },
  extends: [
    'plugin:vue/vue3-essential',
    'standard'
  ],
  overrides: [
  ],
  parserOptions: {
    ecmaVersion: 'latest',
    sourceType: 'module'
  },
  plugins: [
    'vue'
  ],
  rules: {
  }
}
```

指定脚本运行环境，支持多种运行环境

设置继承基础配置中的已启用的规则

设置解析器，例如ECMAScript版本和模块类型

第三方插件

具体规则配置

# 04 使用ESLint

```
var hasBarProperty = foo.hasOwnProperty("bar");

var isPrototypeOfBar = foo.isPrototypeOf(bar);

var barIsEnumerable = foo.propertyIsEnumerable("bar");
```

demo.js

# 04 使用ESLint

## 检验代码

```
PS C:\web-projects\www-site\nodejs-app\ESLint> eslint demo.js

C:\web-projects\www-site\nodejs-app\ESLint\demo.js
  1:1   warning  Unexpected var, use let or const instead                                 no-var
  1:5   error    'hasBarProperty' is assigned a value but never used                      no-unused-vars
  1:22  error    'foo' is not defined                                                     no-undef
  1:26  error    Do not access Object.prototype method 'hasOwnProperty' from target object  no-prototype-builtins
  2:1   warning  Unexpected var, use let or const instead                                 no-var
  2:5   error    'isPrototypeOfBar' is assigned a value but never used                    no-unused-vars
  2:24  error    'foo' is not defined                                                     no-undef
  2:28  error    Do not access Object.prototype method 'isPrototypeOf' from target object  no-prototype-builtins
  2:42  error    'bar' is not defined                                                     no-undef
  3:1   warning  Unexpected var, use let or const instead                                 no-var
  3:5   error    'barIsEnumerable' is assigned a value but never used                     no-unused-vars
  3:23  error    'foo' is not defined                                                     no-undef
  3:27  error    Do not access Object.prototype method 'propertyIsEnumerable' from target object  no-prototype-builtins

✖ 13 problems (10 errors, 3 warnings)
  0 errors and 3 warnings potentially fixable with the `--fix` option.
```

# 04 使用ESLint

检验代码，使用自动fix

```
PS C:\web-projects\www-site\nodejs-app\ESLint> eslint demo.js --fix

C:\web-projects\www-site\nodejs-app\ESLint\demo.js
  1:7   error  'hasBarProperty' is assigned a value but never used                 no-unused-vars
  1:24  error  'foo' is not defined                                                no-undef
  1:28  error  Do not access Object.prototype method 'hasOwnProperty' from target object       no-prototype-builtins
  2:7   error  'isPrototypeOfBar' is assigned a value but never used               no-unused-vars
  2:26  error  'foo' is not defined                                                no-undef
  2:30  error  Do not access Object.prototype method 'isPrototypeOf' from target object        no-prototype-builtins
  2:44  error  'bar' is not defined                                                no-undef
  3:7   error  'barIsEnumerable' is assigned a value but never used                no-unused-vars
  3:25  error  'foo' is not defined                                                no-undef
  3:29  error  Do not access Object.prototype method 'propertyIsEnumerable' from target object  no-prototype-builtins

✖ 10 problems (10 errors, 0 warnings)
```

# 04 使用ESLint

配置规则（告警级别）

1. "off" or 0 - 关闭规则
2. “warn” or 1 – 将规则视为一个警告（不会影响退出码）
3. "error" or 2 - 将规则视为一个错误 (退出码为1)

```
rules: {
    'no-unused-vars': 'off',
    'no-undef': 'warn'
}
```

# 04 使用ESLint

```
PS C:\web-projects\www-site\nodejs-app\ESLint> eslint demo.js --fix

C:\web-projects\www-site\nodejs-app\ESLint\demo.js
  1:24  warning  'foo' is not defined                                          no-undef
  1:28  error    Do not access Object.prototype method 'hasOwnProperty' from target object       no-prototype-builtins
  2:26  warning  'foo' is not defined                                          no-undef
  2:30  error    Do not access Object.prototype method 'isPrototypeOf' from target object        no-prototype-builtins
  2:44  warning  'bar' is not defined                                          no-undef
  3:25  warning  'foo' is not defined                                          no-undef
  3:29  error    Do not access Object.prototype method 'propertyIsEnumerable' from target object  no-prototype-builtins

✖ 7 problems (3 errors, 4 warnings)
```

# 04 使用ESLint

配置规则（全局变量）

1. globals 配置属性设置为一个对象
2. 该对象包含开发人员希望使用的每个全局变量
3. “writable” 表示允许重写变量， "readonly" 不允许重写变量

```
globals: {
  bar: 'readonly'
},
```

# 04 使用ESLint

```
PS C:\web-projects\www-site\nodejs-app\ESLint> eslint demo.js --fix

C:\web-projects\www-site\nodejs-app\ESLint\demo.js
  1:24  warning  'foo' is not defined                                          no-undef
  1:28  error    Do not access Object.prototype method 'hasOwnProperty' from target object       no-prototype-builtins
  2:26  warning  'foo' is not defined                                          no-undef
  2:30  error    Do not access Object.prototype method 'isPrototypeOf' from target object        no-prototype-builtins
  3:25  warning  'foo' is not defined                                          no-undef
  3:29  error    Do not access Object.prototype method 'propertyIsEnumerable' from target object  no-prototype-builtins

✖ 6 problems (3 errors, 3 warnings)
```

# 04 使用ESLint

除了可以使用配置文件去配置代码规则，还可以使用代码注释的方式进行配置。

```
1  /* eslint-disable-next-line no-prototype-builtins, no-undef */
2  const hasBarProperty = foo.hasOwnProperty('bar')
3  const isPrototypeOfBar = foo.isPrototypeOf(bar)
4  const barIsEnumerable = foo.propertyIsEnumerable('bar') // eslint-disable-line no-prototype-builtins, no-undef
```

```
PS C:\web-projects\www-site\nodejs-app\ESLint> eslint demo.js --fix

C:\web-projects\www-site\nodejs-app\ESLint\demo.js
  3:26  warning  'foo' is not defined                                                 no-undef
  3:30  error    Do not access Object.prototype method 'isPrototypeOf' from target object  no-prototype-builtins

✖ 2 problems (1 error, 1 warning)
```

# 04 使用ESLint

```
/* eslint-disable-next-line no-prototype-builtins, no-undef */
const hasBarProperty = foo.hasOwnProperty('bar')
/* eslint-disable-next-line no-undef */
const isPrototypeOfBar = Object.prototype.isPrototypeOf.call(foo, bar)
// const isPrototypeOfBar = foo.isPrototypeOf(bar)
const barIsEnumerable = foo.propertyIsEnumerable('bar') // eslint-disable-line no-prototype-builtins, no-undef
```

```
  globals: {
    bar: 'readonly'
  },
  rules: {
    'no-unused-vars': 0,
    'no-undef': 1
  }
```

```
PS C:\web-projects\www-site\nodejs-app\ESLint> eslint demo.js --fix
PS C:\web-projects\www-site\nodejs-app\ESLint> eslint demo.js
```

# 05 ESLint发展趋势

- 成为事实上的Web前端标准
- 与开发框架高度结合