01_Vue2#创建示例项目👈

创建时间：`2022年07月26日 16:25:33`

更新时间：`2023年08月15日 21:28:33`

示例项目：

- 📎vuex3-demo.zip

> Vue官方提供了Vue的核心npm包和vue-cli
>
> - 这里我们将使用vue-cli初始化一个Vue项目

一：安装Vue-Cli

- 此处为语雀内容卡片，点击链接查看：https://www.yuque.com/u22384616/ncmk42/ptqc9q93pz9l07qo?view=doc_embed
- 此处为语雀内容卡片，点击链接查看：https://www.yuque.com/u22384616/ncmk42/cc0tes?view=doc_embed

> npm view @vue/cli

Bash | 复制代码

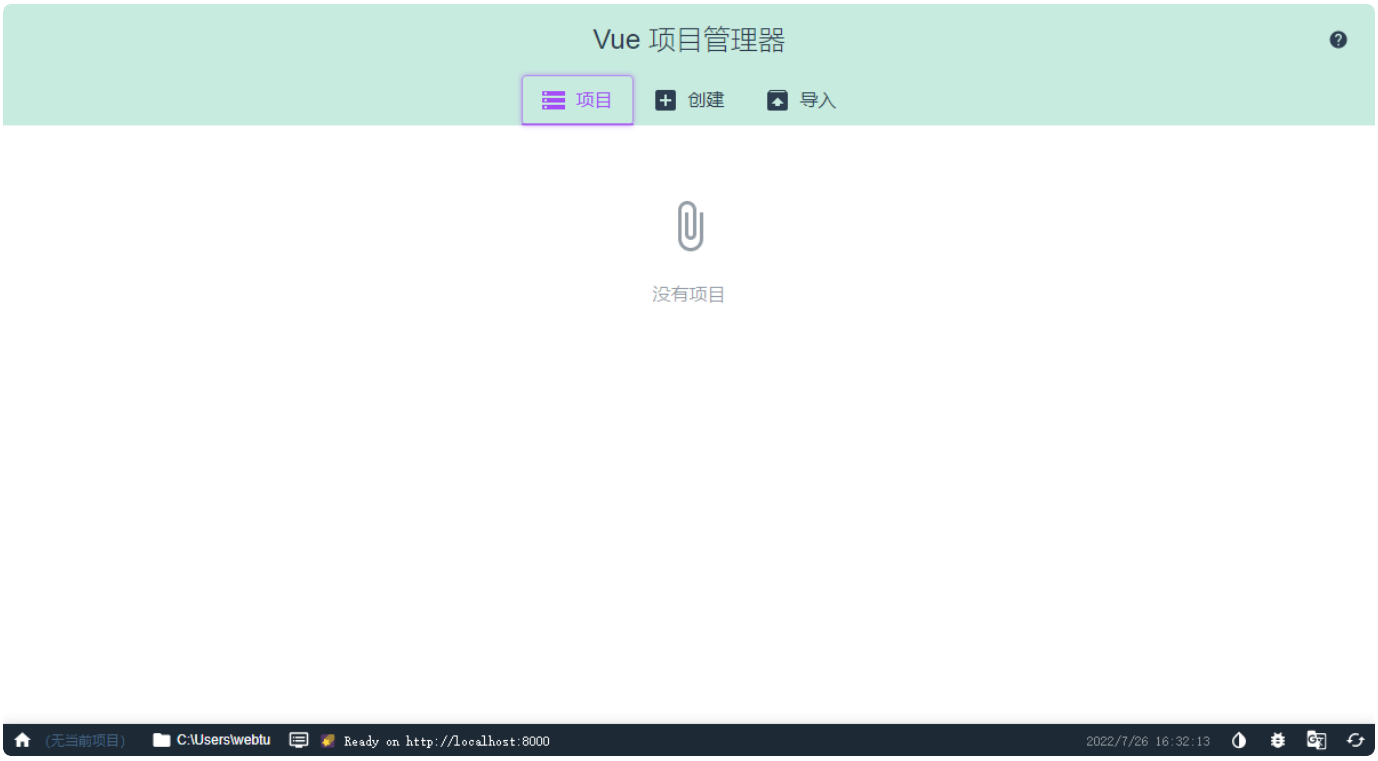
```
C:\Users\webtu\Desktop>npm view @vue/cli
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.

@vue/cli@5.0.8 | MIT | deps: 35 | versions: 172
Command line interface for rapid Vue.js development
https://cli.vuejs.org/

bin: vue

dist
.tarball: https://mirrors.cloud.tencent.com/npm/@vue%2fcli/-/cli-5.0.8.tgz
.shasum: 97b2bad9cb331dcffdd4fbe8532bdeacd2441166
.integrity: sha512-c/QKPdC09bYkW22m/boXkLaiz10z0Z2WHZO7zEeNdfSduqyWINZhKc6hVQU3Vk0NXW7BJAd7zWmcUrC8L9TuAA==
.unpackedSize: 158.6 kB

dependencies:
@types/ejs: ^3.0.6                debug: ^4.1.0                     ini: ^2.0.0
@types/inquirer: ^8.1.3           deepmerge: ^4.2.2                 inquirer: ^8.0.0
@vue/cli-shared-utils: ^5.0.8     download-git-repo: ^3.0.2         isbinaryfile: ^4.0.6
@vue/cli-ui-addon-webpack: ^5.0.8 ejs: ^3.1.6                       javascript-stringify: ^2.0.1
@vue/cli-ui-addon-widgets: ^5.0.8 envinfo: ^7.7.4                   js-yaml: ^4.0.0
@vue/cli-ui: ^5.0.8               fs-extra: ^9.1.0                  leven: ^3.1.0
boxen: ^5.0.0                     globby: ^11.0.2                   lodash.clonedeep: ^4.5.0
commander: ^7.1.0                 import-global: ^0.1.0             lru-cache: ^6.0.0
(...and 11 more.)

dist-tags:
latest: 5.0.8  next: 5.0.6

published 2 weeks ago by soda <haoqunjiang+npm@gmail.com>

C:\Users\webtu\Desktop>
C:\Users\webtu\Desktop>npm ls -g @vue/cli
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
```

```
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
C:\Users\webtu\AppData\Roaming\npm
`-- @vue/cli@5.0.8
```

> npm view @vue/cli
>
> npm install –g @vue/cli
>
> npm ls –g @vue/cli
>
> vue ui

二：启动vue ui

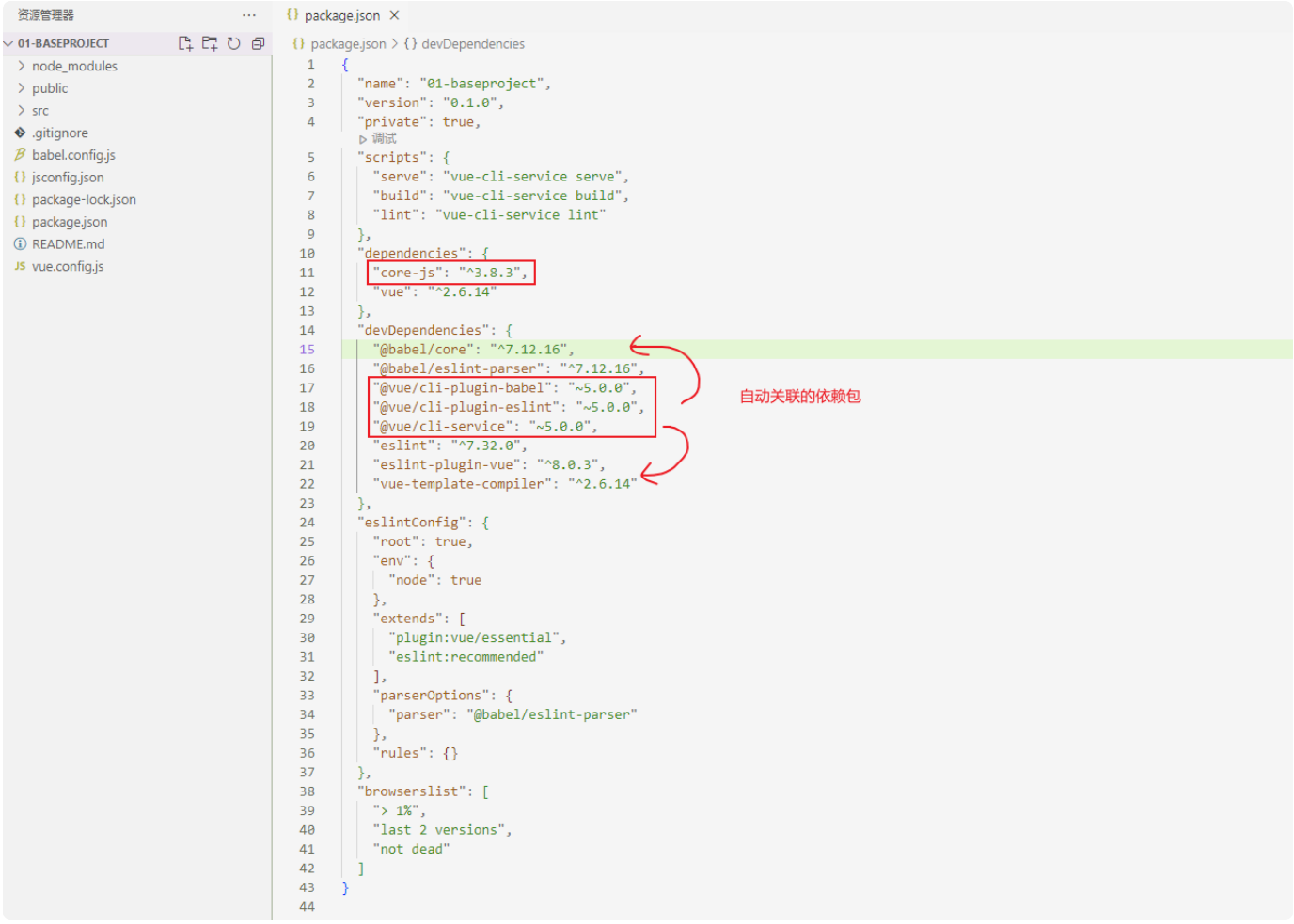
> vue ui



三：创建项目

- @vue/cli-service
  - vue-cli-service serve
  - vue-cli-service build

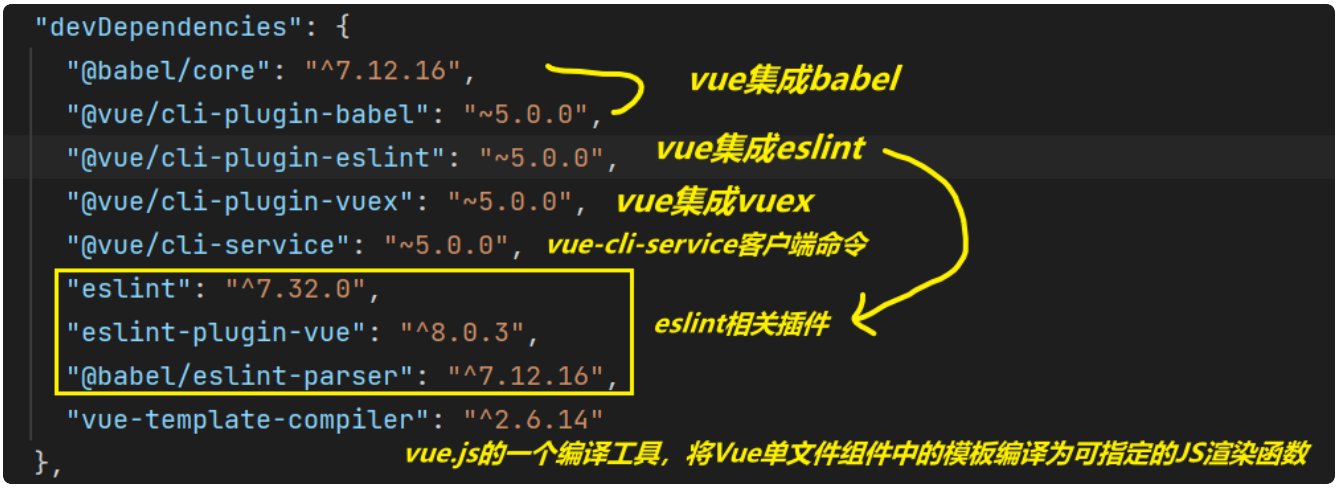
- vue-cli-service lint
  - 自动修复项目下的js文件，如果有插件则会检查插件包含格式的文件

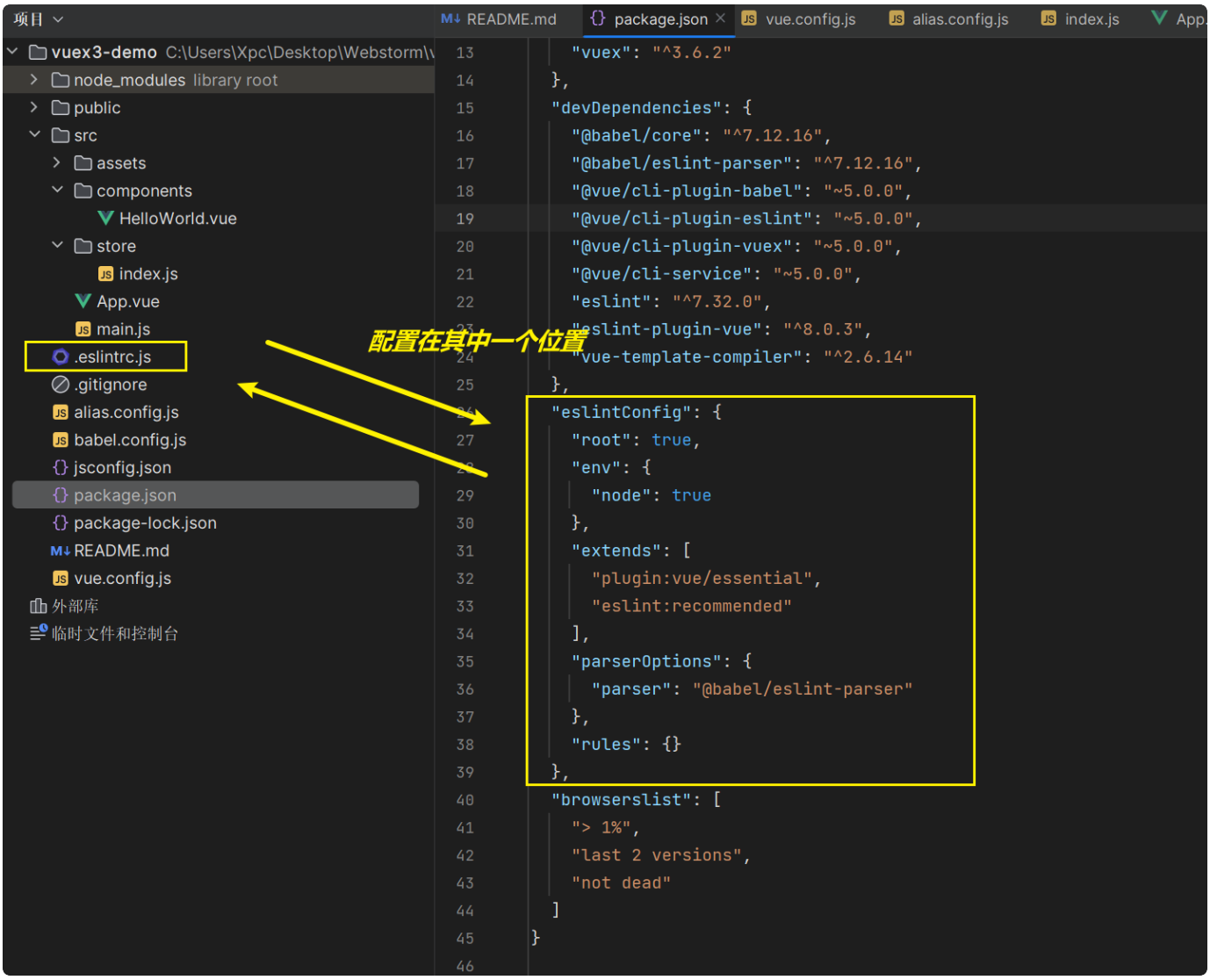


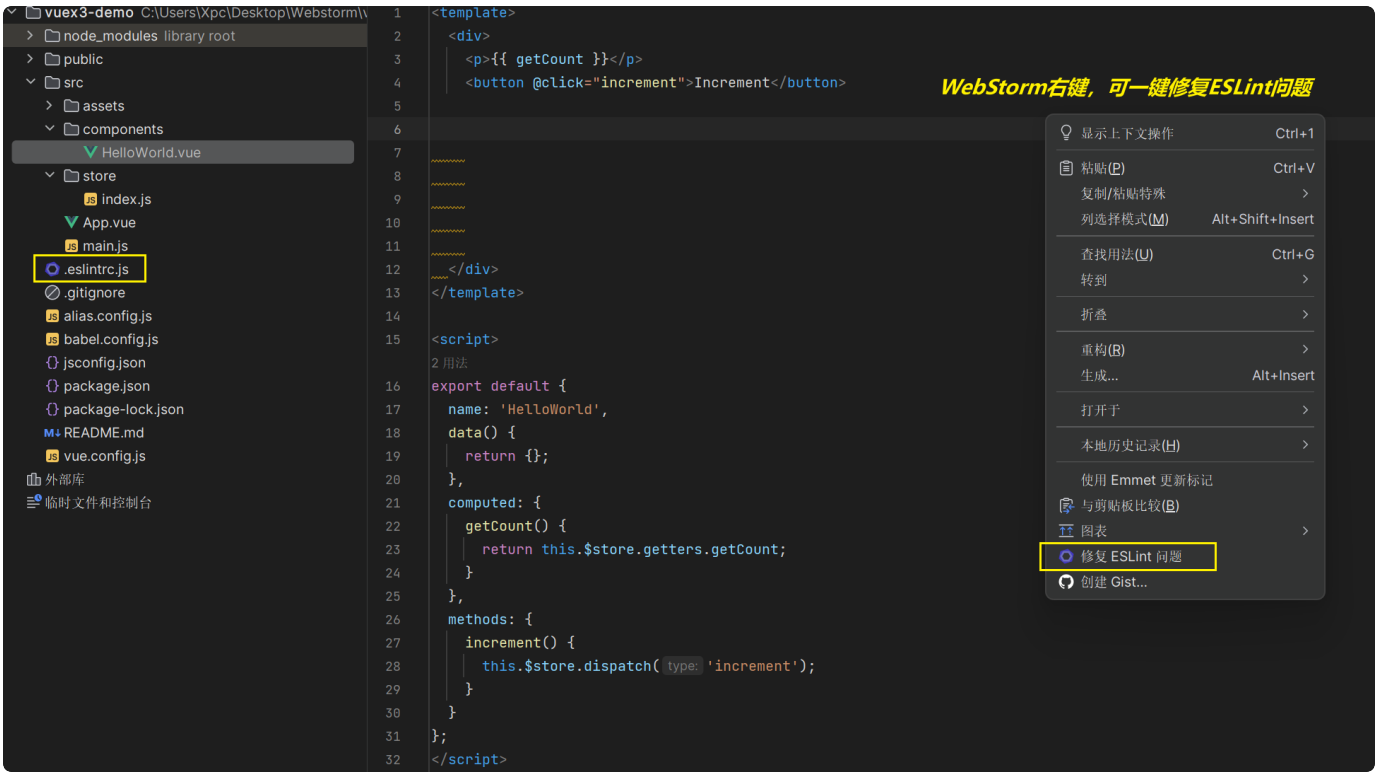
> core-js
>
> - https://cloud.tencent.com/developer/article/1893817

四：配置eslint（✨）

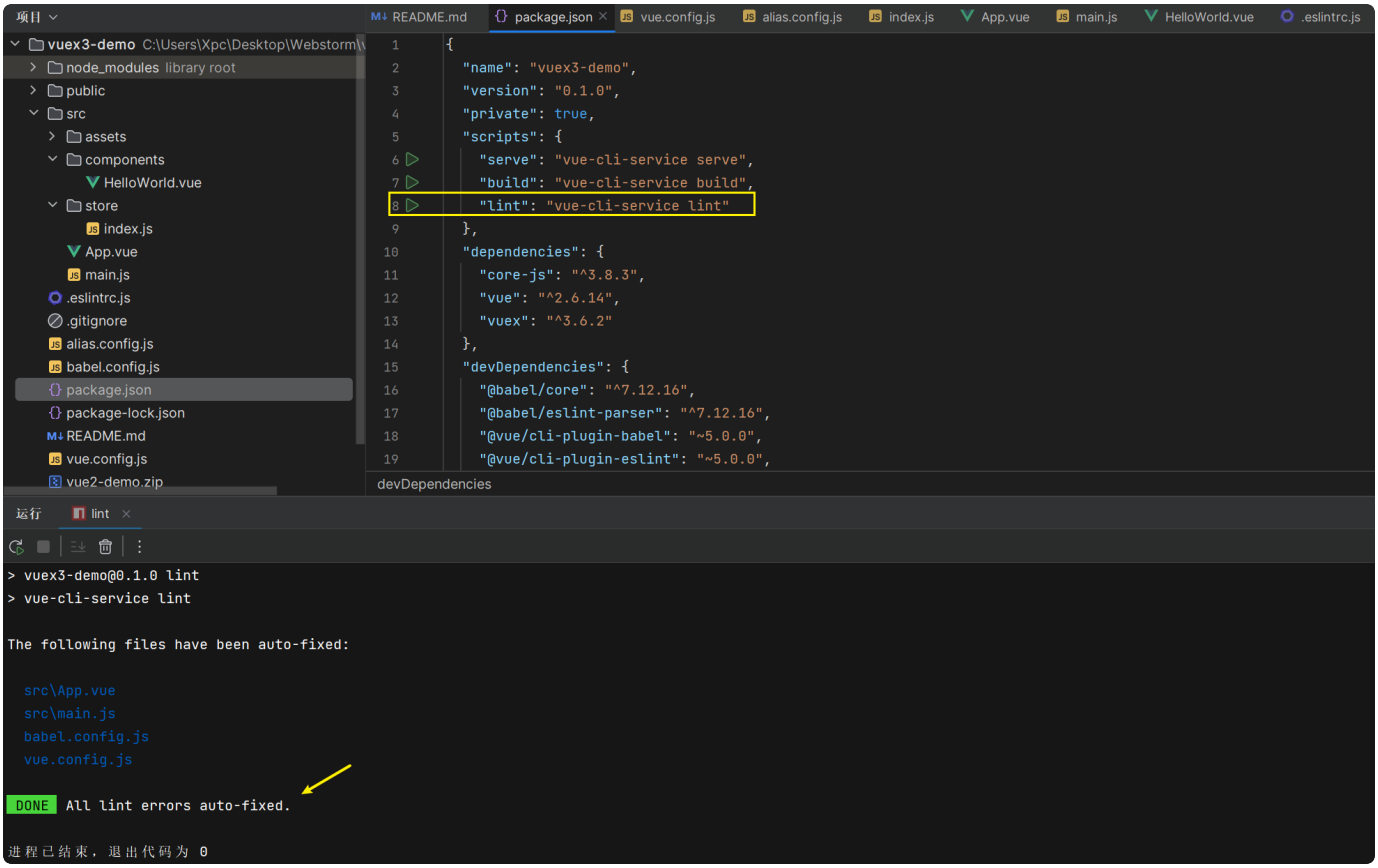
> ✨ `"eslint": "^7.32.0",`
>
> ✨ `"eslint-plugin-vue": "^8.0.3",`
>
> ✨ `"@babel/eslint-parser": "^7.12.16",`


"devDependencies": {
"@babel/core": "^7.12.16",
"@vue/cli-plugin-babel": "~5.0.0",
"@vue/cli-plugin-eslint": "~5.0.0",
"@vue/cli-plugin-vuex": "~5.0.0",
"@vue/cli-service": "~5.0.0",
"eslint": "^7.32.0",
"eslint-plugin-vue": "^8.0.3",
"@babel/eslint-parser": "^7.12.16",
"vue-template-compiler": "^2.6.14"
},
vue集成babel
vue集成eslint
vue集成vuex
vue-cli-service客户端命令
eslint相关插件
vue.js的一个编译工具，将Vue单文件组件中的模板编译为可指定的JS渲染函数


项目
vuex3-demo C:\Users\Xpc\Desktop\Webstorm\v
node_modules library root
public
src
assets
components
HelloWorld.vue
store
index.js
App.vue
main.js
.eslintrc.js
.gitignore
alias.config.js
babel.config.js
jsconfig.json
package.json
package-lock.json
README.md
vue.config.js
外部库
临时文件和控制台
README.md
package.json
vue.config.js
alias.config.js
index.js
"vuex": "^3.6.2"
},
"devDependencies": {
"@babel/core": "^7.12.16",
"@babel/eslint-parser": "^7.12.16",
"@vue/cli-plugin-babel": "~5.0.0",
"@vue/cli-plugin-eslint": "~5.0.0",
"@vue/cli-plugin-vuex": "~5.0.0",
"@vue/cli-service": "~5.0.0",
"eslint": "^7.32.0",
"eslint-plugin-vue": "^8.0.3",
"vue-template-compiler": "^2.6.14"
},
"eslintConfig": {
"root": true,
"env": {
"node": true
},
"extends": [
"plugin:vue/essential",
"eslint:recommended"
],
"parserOptions": {
"parser": "@babel/eslint-parser"
},
"rules": {}
},
"browserslist": [
"> 1%",
"last 2 versions",
"not dead"
]
}
配置在其中一个位置



😊一键检测+修复所有ESLint问题



> http://eslint.cn/docs/user-guide/configuring

- 使用单独的配置文件.eslintrc.js

- 将eslint配置写在package.json中

> .eslintrc.js

JavaScript | 复制代码

```javascript
module.exports = {
    root: true,
    env: {
        // 表示在当前代码环境中启用了 Node.js 相关的全局变量和规则。
        //   → 设置为 { node: true } 可以让 ESLint 知道代码被执行在 Node.js 的环境中，从而正确地处理 Node.js 相关的全局变量和规则。
        //   → 一个环境定义了一个一组预定义的全局变量。比如node环境的全局变量是global，但是浏览器环境的全局对象是window。
        node: true
    },

    // extends用于扩展和继承其他 ESLint 配置。
    extends: [
        // 表示扩展了名为 vue/essential 的插件。这个插件提供了一组用于 Vue.js 项目的基本规则。通过扩展该插件，可以启用适用于 Vue.js 项目的默认规则。
        "plugin:vue/essential",
        // 表示继承了名为 eslint:recommended 的配置。这是 ESLint 官方推荐的一组规则，涵盖了一系列常见的 JavaScript 最佳实践和代码质量规则。
        // "eslint:recommended" 规则集加载并启用了一组常见的规则，但在 rules 中你仍然需要显式地根据自己的需求定义和配置规则的行为。
        "eslint:recommended"
    ],
    parserOptions: {
        "parser": "@babel/eslint-parser"
    },
    rules: {
        // 允许代码中使用console
        "no-console": "off",

        // 空行最多不能超过2行
        'no-multiple-empty-lines': [1, { 'max': 2 }],

        // Vue ESLint 插件提供的规则: 禁用对无效结束标签的解析错误的检查。即当出现无效结束标签时，ESLint 不会将其视为解析错误。
        'vue/no-parsing-error': [2, {
            'x-invalid-end-tag': false
        }],

        // 禁止对const变量重新赋值
        'no-const-assign': 2,

        // 变量声明后却不使用检测[0关闭，1警告，2错误]
        'no-unused-vars': [0, {
            'vars': 'all',
            'args': 'none'
```

```
        }],
        'quotes': [2, 'single'],     // 单引号
        'semi': [2, 'always'],       // 强制分号
        'indent': ['error', 2],      // 缩进量('indent': [2, 2]效果是一致的)
        'no-var': 2,                 // 禁用var，用let和const代替
        'camelcase': 2,              // 强制驼峰法命名
        'eqeqeq': 1,                 // 要求使用 === 和 !== 代替 == 和 != 操作符
        'no-eq-null': 2              // 禁止对null使用==或!=运算符,相应地，推荐使用严格相等运算符（=== 或 !==）来进行比较。

    }
};
```

🤔 `root: true`

- `root: true`：表示该配置文件的作用范围为根目录。如果设置为 `true`，则 ESLint 将在当前配置文件所在的目录中停止查找其他配置文件。
- 如果将 `root` 配置设置为 `false`，它的含义就是将配置文件的作用范围扩展到父目录，而不仅仅是当前配置文件所在的目录。

  当 `root` 配置为 `false` 时，ESLint 会继续向上查找，并在父目录中寻找其他的 ESLint 配置文件。这样可以允许你在子目录中使用不同的 ESLint 配置来覆盖或扩展父目录的配置。

  总结来说，`root: false` 表示配置文件的作用范围会扩展到父目录，以便允许在子目录中使用不同的 ESLint 配置。

🤔 `'parser': '@babel/eslint-parser'`

- 在 ESLint 配置文件中，`parserOptions` 部分用于配置解析器的行为和选项。在给定的代码示例中，`parserOptions` 设置了解析器为 `@babel/eslint-parser`。

  `@babel/eslint-parser` 是一个由 Babel 提供的用于解析 JavaScript 代码的解析器。它能够解析使用了 Babel 语法扩展的代码，包括支持最新的 ECMAScript 特性和其他非标准的语法。

  通过将 `parserOptions` 中的 `parser` 属性设置为 `@babel/eslint-parser`，你告诉 ESLint 使用这个解析器来解析你的 JavaScript 代码。

  这样配置解析器的好处是，它允许你在 ESLint 中使用最新的 JavaScript 语法和 ECMAScript 特性，即使这些特性在当前运行环境中不被原生支持。Babel 解析器能够将这些语法扩展转换为标准的 ECMAScript 代码，以便进行静态代码分析和代码质量检查。

  需要注意的是，使用 `@babel/eslint-parser` 解析器需要确保你的项目中安装了相应的依赖。你需要安装 `@babel/core`、`@babel/eslint-parser` 和其他相关的 Babel 插件，以便正确配置和运行解析器。

  综上所述，通过在 `parserOptions` 中设置 `parser` 为 `@babel/eslint-parser`，你能够使用 Babel 解析器来解析你的 JavaScript 代码，并使用最新的 ECMAScript 特性和语法扩展进行静态代码分析和代码质量检查。

```
"dependencies": {
  "core-js": "^3.8.3",
  "vue": "^2.6.14",
  "vuex": "^3.6.2"
},
"devDependencies": {
  "@babel/core": "^7.12.16",
  "@babel/eslint-parser": "^7.12.16",
  "@vue/cli-plugin-babel": "~5.0.0",
  "@vue/cli-plugin-eslint": "~5.0.0",
  "@vue/cli-plugin-vuex": "~5.0.0",
  "@vue/cli-service": "~5.0.0",
  "eslint": "^7.32.0",
  "eslint-plugin-vue": "^8.0.3",
  "vue-template-compiler": "^2.6.14"
},
```

- ESLint 是一个静态代码分析工具，它主要用于检查代码的质量和符合性，而不会对代码进行转换。它可以检查代码格式、潜在的bug、代码风格问题等，但不会对代码进行转换或编译。

  ESLint 的主要功能是根据预定义的规则和配置文件对代码进行检查。它会在解析 JavaScript 代码之后，通过分析抽象语法树（AST）来检查代码是否符合规则。

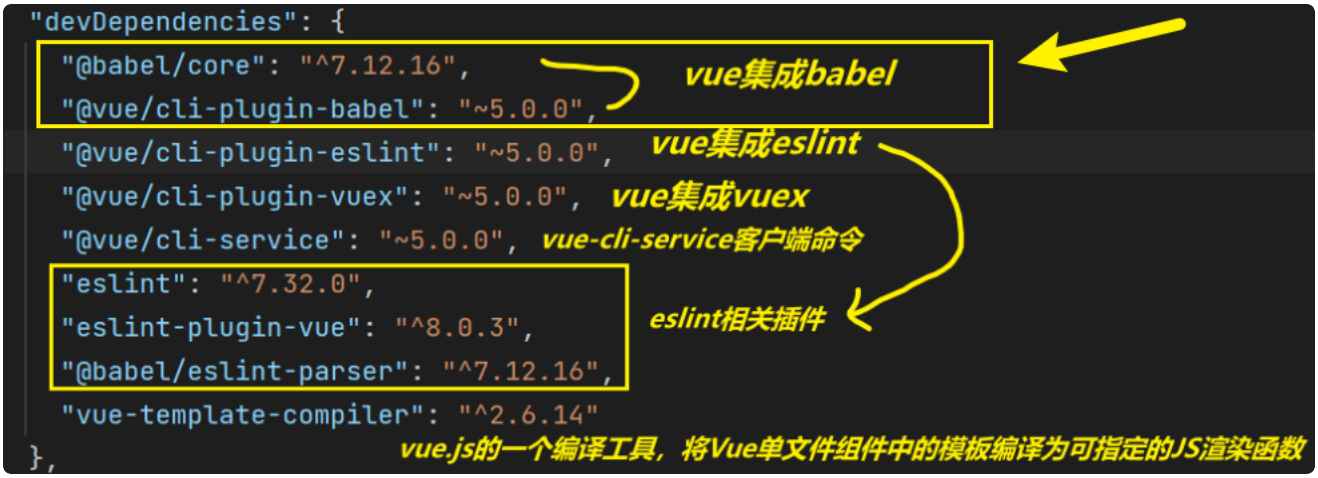
  虽然 ESLint 可以指定一些规则来帮助发现一些常见的错误和潜在的问题，但它并不会对代码进行任何转换工作。如果你需要对代码进行转换或编译，你可以使用 Babel、TypeScript、Webpack 等工具来处理代码，这些工具可以将代码转换为不同的版本或其他格式。

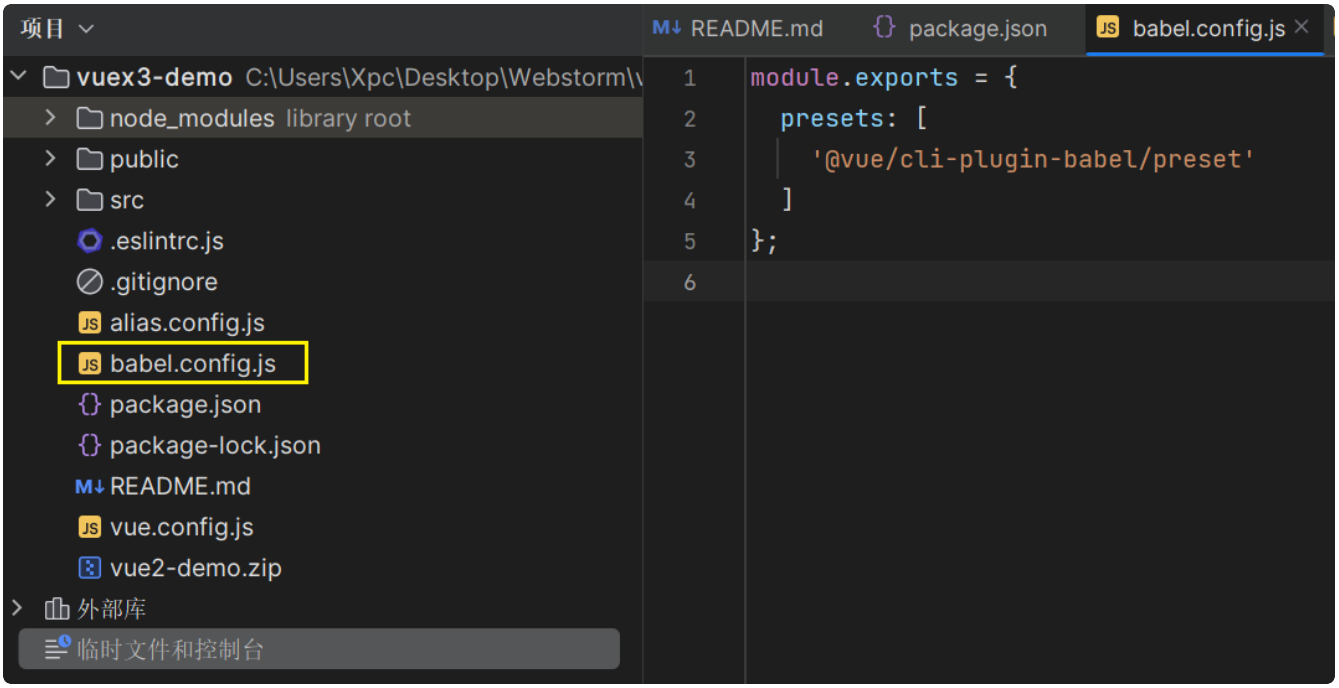
  总之，ESLint 是一个用于静态代码分析的工具，它主要用于检查代码的质量和符合性，而不会对代码进行转换。它能够帮助发现潜在的错误、遵循代码风格规范等，并提供规则来指导开发者编写更好的代码。转换和编译代码通常需要使用其他工具来完成。

五、`babel.config.js`

> *babel.config.js 文件用于配置 Babel 的转译行为和选项。*
>
> *Babel 是一个 JavaScript 编译器，用于将新版本的 JavaScript 代码转译成旧版本的 JavaScript 代码，以便在较旧的浏览器或环境中运行。*

- ✨ `"@babel/core": "^7.12.16",`
- ✨ `"@vue/cli-plugin-babel": "~5.0.0",`


"devDependencies": {
"@babel/core": "^7.12.16",
"@vue/cli-plugin-babel": "~5.0.0",
vue集成babel
"@vue/cli-plugin-eslint": "~5.0.0",
vue集成eslint
"@vue/cli-plugin-vuex": "~5.0.0",
vue集成vuex
"@vue/cli-service": "~5.0.0",
vue-cli-service客户端命令
"eslint": "^7.32.0",
"eslint-plugin-vue": "^8.0.3",
"@babel/eslint-parser": "^7.12.16",
eslint相关插件
"vue-template-compiler": "^2.6.14"
},
vue.js的一个编译工具，将Vue单文件组件中的模板编译为可指定的JS渲染函数


项目
vuex3-demo C:\Users\Xpc\Desktop\Webstorm\
node_modules library root
public
src
.eslintrc.js
.gitignore
alias.config.js
babel.config.js
package.json
package-lock.json
README.md
vue.config.js
vue2-demo.zip
外部库
临时文件和控制台
README.md
package.json
babel.config.js
module.exports = {
presets: [
'@vue/cli-plugin-babel/preset'
]
};

`babel.config.js` 文件用于配置 Babel 的转译行为和选项。

Babel 是一个 JavaScript 编译器，用于将新版本的 JavaScript 代码转译成旧版本的 JavaScript 代码，以便在较旧的浏览器或环境中运行。Babel 可以转译的内容包括最新的 ECMAScript 语法和特性，以及各种其他的 JavaScript 扩展、插件和转换工具等。

`babel.config.js` 文件是用于配置 Babel 的主要配置文件之一（另一个常用的是 `.babelrc` 文件）。它通常位于项目的根目录下，用于指定 Babel 的转译规则和插件。

在 `babel.config.js` 文件中，你可以配置 Babel 的预设（presets）和插件（plugins），以及其他选项和设置。预设是一组特定的规则、插件和配置的集合，用于处理特定的 JavaScript 语法和特性。插件是个别的、可选的转译功能或工具，用于处理特定的代码转换需求。

通过编辑 `babel.config.js` 文件，你可以配置 Babel 的转译行为，包括指定要支持的 ECMAScript 版本、转译语法、插件等。你还可以根据项目的需要进行自定义配置，以满足不同的转译需求。

需要注意的是，如果你的项目使用了 `babel.config.js` 文件作为 Babel 的配置文件，并定义了转译规则和插件，那么 Babel 将会完全按照这个配置文件的规则进行转译，忽略其他可能的 `.babelrc` 或者 `.babelrc.js` 文件。

综上所述，`babel.config.js` 文件用于配置 Babel 的转译行为和选项。通过编辑这个文件，你可以定义 Babel 的预设、插件和其他选项来控制 JavaScript 代码的转译过程，以便运行在不同版本的浏览器或环境中。

babel.config.js解析🔽

JavaScript | 复制代码

```
module.exports = {
  presets: [
    '@vue/cli-plugin-babel/preset'
  ]
};
```

这是一个使用 `babel.config.js` 文件配置 Babel 的示例。在这个示例中，配置了一个使用 `@vue/cli-plugin-babel/preset` 预设的 `presets` 数组。

`@vue/cli-plugin-babel/preset` 是一个由 Vue CLI 提供的预设，用于配置 Babel 来转译 Vue.js 项目的代码。它集成了常见的转译规则和插件，以便于开发者编写和运行符合 Vue.js 标准的 JavaScript 代码。

通过将 `@vue/cli-plugin-babel/preset` 添加到 `presets` 数组中，告诉 Babel 使用这个预设来处理 Vue.js 项目的代码转译。使用预设可以方便地处理 Vue.js 中的模板编译、单文件组件以及其他相关的语法扩展，以确保这些功能在浏览器中正常运行。

此外，你还可以在 `babel.config.js` 文件中添加其他的配置选项，以满足你的项目需求。你可以添加额外的预设或插件，或根据需要进行自定义配置。

需要注意的是，为了让上述配置生效，你需要确保已安装了相关的依赖，包括 `@vue/cli-plugin-babel` 和其他与 Vue.js 相关的 Babel 插件。

综上所述，将 `@vue/cli-plugin-babel/preset` 添加到 `presets` 数组中，可以使用 `babel.config.js` 文件配置 Babel 来处理 Vue.js 项目的代码转译。这样可以确保 Vue.js 的相关功能在浏览器中正常运行，并简化了配置过程。

六、 `package.json` x2

```
"devDependencies": {
  "@babel/core": "^7.12.16",
  "@vue/cli-plugin-babel": "~5.0.0",
  "@vue/cli-plugin-eslint": "~5.0.0",
  "@vue/cli-plugin-vuex": "~5.0.0",
  "@vue/cli-service": "~5.0.0",
  "eslint": "^7.32.0",
  "eslint-plugin-vue": "^8.0.3",
  "@babel/eslint-parser": "^7.12.16",
  "vue-template-compiler": "^2.6.14"
},
```

vue集成babel

vue集成eslint

vue集成vuex

vue-cli-service客户端命令

eslint相关插件

vue.js的一个编译工具，将Vue单文件组件中的模板编译为可指定的JS渲染函数

版本一（不再推荐）

JSON | 复制代码

```json
{
  "name": "vuex3-demo",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint"
  },
  "dependencies": {
    "core-js": "^3.8.3",
    "vue": "^2.6.14",
    "vuex": "^3.6.2"
  },
  "devDependencies": {
    "@babel/core": "^7.12.16",
    "@babel/eslint-parser": "^7.12.16",
    "@vue/cli-plugin-babel": "~5.0.0",
    "@vue/cli-plugin-eslint": "~5.0.0",
    "@vue/cli-plugin-vuex": "~5.0.0",
    "@vue/cli-service": "~5.0.0",
    "eslint": "^7.32.0",
    "eslint-plugin-vue": "^8.0.3",
    "vue-template-compiler": "^2.6.14"
  },
  "eslintConfig": {
    "root": true,
    "env": {
      "node": true
    },
    "extends": [
      "plugin:vue/essential",
      "eslint:recommended"
    ],
    "parserOptions": {
      "parser": "@babel/eslint-parser"
    },
    "rules": {}
  },
  "browserslist": [
    "> 1%",
    "last 2 versions",
    "not dead"
  ]
}
```

```
46
```

版本二✔

JSON | 复制代码

```json
{
  "name": "vuex3-demo",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint"
  },
  "dependencies": {
    "vue": "^2.6.14",
    "vuex": "^3.6.2"
  },
  "devDependencies": {
    "@babel/core": "^7.12.16",
    "@babel/preset-env": "^7.12.16",
    "@vue/cli-plugin-babel": "~5.0.0",
    "@vue/cli-plugin-eslint": "~5.0.0",
    "@vue/cli-plugin-vuex": "~5.0.0",
    "@vue/cli-service": "~5.0.0",
    "eslint": "^7.32.0",
    "eslint-plugin-vue": "^8.0.3",
    "@babel/eslint-parser": "^7.12.16",
    "vue-template-compiler": "^2.6.14"
  },
  "browserslist": [
    "> 1%",
    "last 2 versions",
    "not dead"
  ]
}

```

问题一：🤔 `browserlist`？

> *以 Babel 为例，它是怎么判断项目是否需要做降级处理的呢，答案就是通过 browserslist 查询出需要支持的浏览器列表，然后根据这个列表来做判断。*

问题二：🤔 `@vue/cli-plugin-eslint`？

> *注入vue-cli-service lint命令*

> - https://cli.vuejs.org/core-plugins/eslint.html

问题三：🤔 `core-js`？

- babel和core-js结合使用，插件会自动调用core-js进行JavaScript语法转换。

> ⚠️⚠️⚠️
>
> *需要注意的是，引入 core-js 库会增加打包体积，因此我们可以使用按需引入的方式来避免不必要的代码冗余。*
>
> *`Vue CLI 3.x` 提供了 `@babel/preset-env` 和 `@vue/cli-plugin-babel/preset` 预设配置，可以根据目标浏览器或运行时环境智能地进行特性填充，从而优化打包体积。*

babel.config.js新版 JavaScript 复制代码

```
module.exports = {
  presets: [
    '@vue/cli-plugin-babel/preset',
    [
      '@babel/preset-env',
      {
        useBuiltIns: 'entry',
        corejs: 3,
      },
    ],
  ],
};
```

> ⚠️⚠️⚠️
>
> *如果你选择了"useBuiltIns": "entry"选项，那么在编译代码时，它将自动根据你的代码中使用的特性自动引入相关的core-js polyfill。这样做可以减少打包体积，仅按需引入所需的polyfill。*
>
> *因此，当你使用@babel/preset-env和"useBuiltIns": "entry"选项时，你无需显式地在项目的dependencies中引入core-js库。它会根据代码需求自动引入所需的polyfill。*

问题四：🤔 `vue-template-compiler?`

`vue-template-compiler` 是 Vue.js 的一个编译工具，它的主要作用是将 Vue 单文件组件中的模板编译为可执行的 JavaScript 渲染函数。

在 Vue 单文件组件中，我们可以使用模板语法来定义组件的视图结构。模板中包含了一些特定的语法和指令，用于描述组件的渲染逻辑、数据绑定、事件处理等。然而，浏览器无法直接理解和执行这些模板语法，因此需要将其编译为浏览器可以执行的 JavaScript 代码。

`vue-template-compiler` 提供了一个编译器，它可以将 Vue 单文件组件中的模板编译为可执行的 JavaScript 渲染函数。这些渲染函数是 Vue.js 内部用于实现虚拟 DOM 渲染的核心机制。通过编译模板为渲染函数，Vue.js 可以将组件的模板转换为虚拟 DOM，并进行数据绑定和渲染等工作。

`vue-template-compiler` 的主要功能包括：

1. **模板编译**：将 Vue 单文件组件中的模板编译为 JavaScript 渲染函数。这些渲染函数将被 Vue.js 内部使用，用于构建虚拟 DOM 并进行组件的渲染。

2. **数据绑定**：编译过程中，`vue-template-compiler` 会解析模板中的数据绑定表达式，并将其转换为响应式的数据绑定关系。这样，在组件渲染时，数据的变化会触发渲染函数的重新执行，从而实现响应式的 UI 更新。

3. **渲染性能优化**：`vue-template-compiler` 会对模板进行静态分析，找出那些在每次渲染中都保持不变的部分，并进行静态化处理。这样可以提升渲染性能，减少不必要的重复计算和渲染。

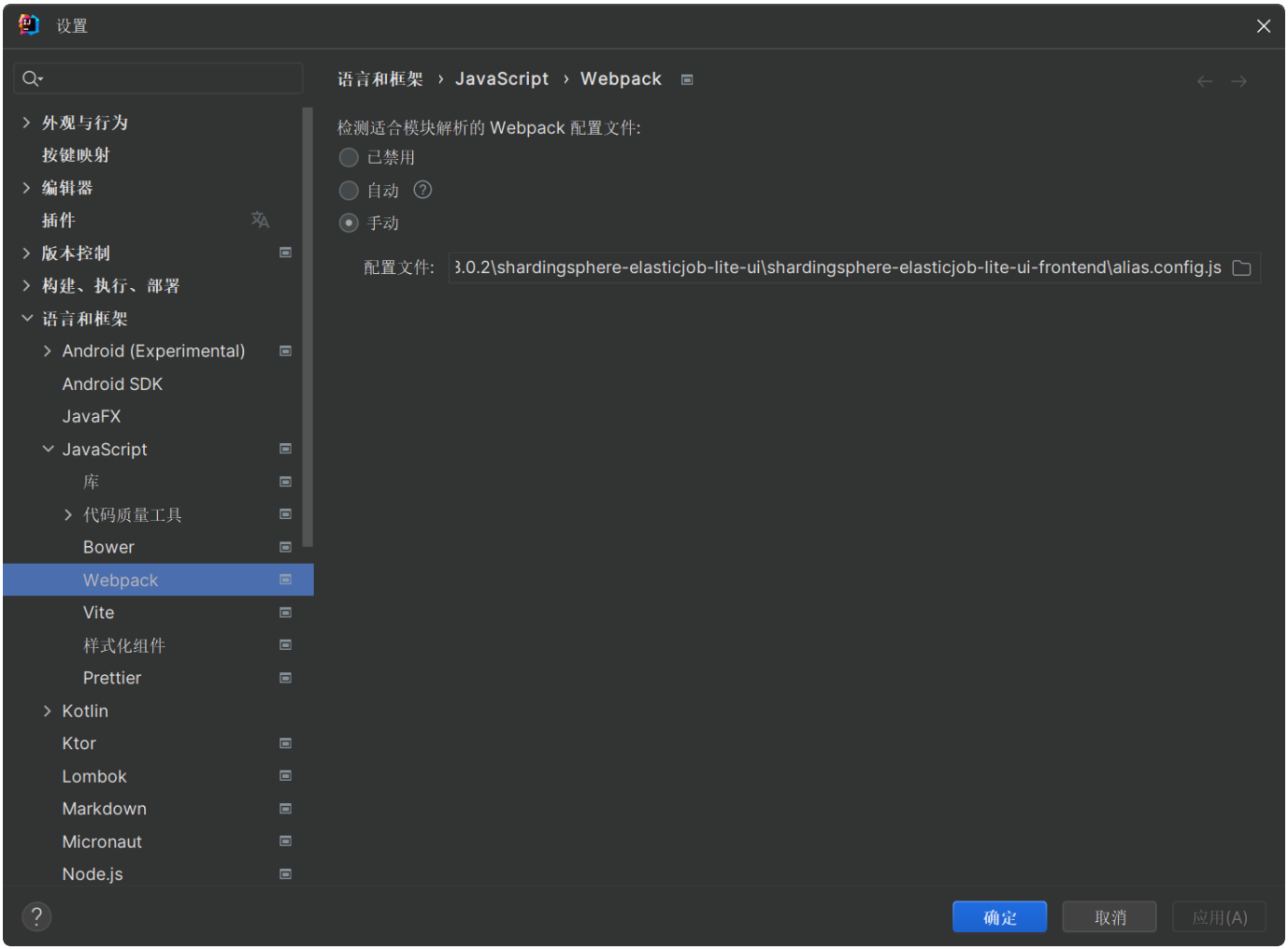
需要注意的是，`vue-template-compiler` 是作为 Vue.js 的一个独立包发布的。它通常作为开发依赖项使用，用于构建和编译 Vue 单文件组件。在生产环境中，你不需要将 `vue-template-compiler` 包含在最终的构建文件中。

综上所述，`vue-template-compiler` 的作用是将 Vue 单文件组件中的模板编译为可执行的 JavaScript 渲染函数，用于构建虚拟 DOM 并实现组件的渲染和响应式数据绑定。它是 Vue.js 生态系统中重要的一部分，用于提供高效的组件渲染和数据驱动的 UI 更新机制。

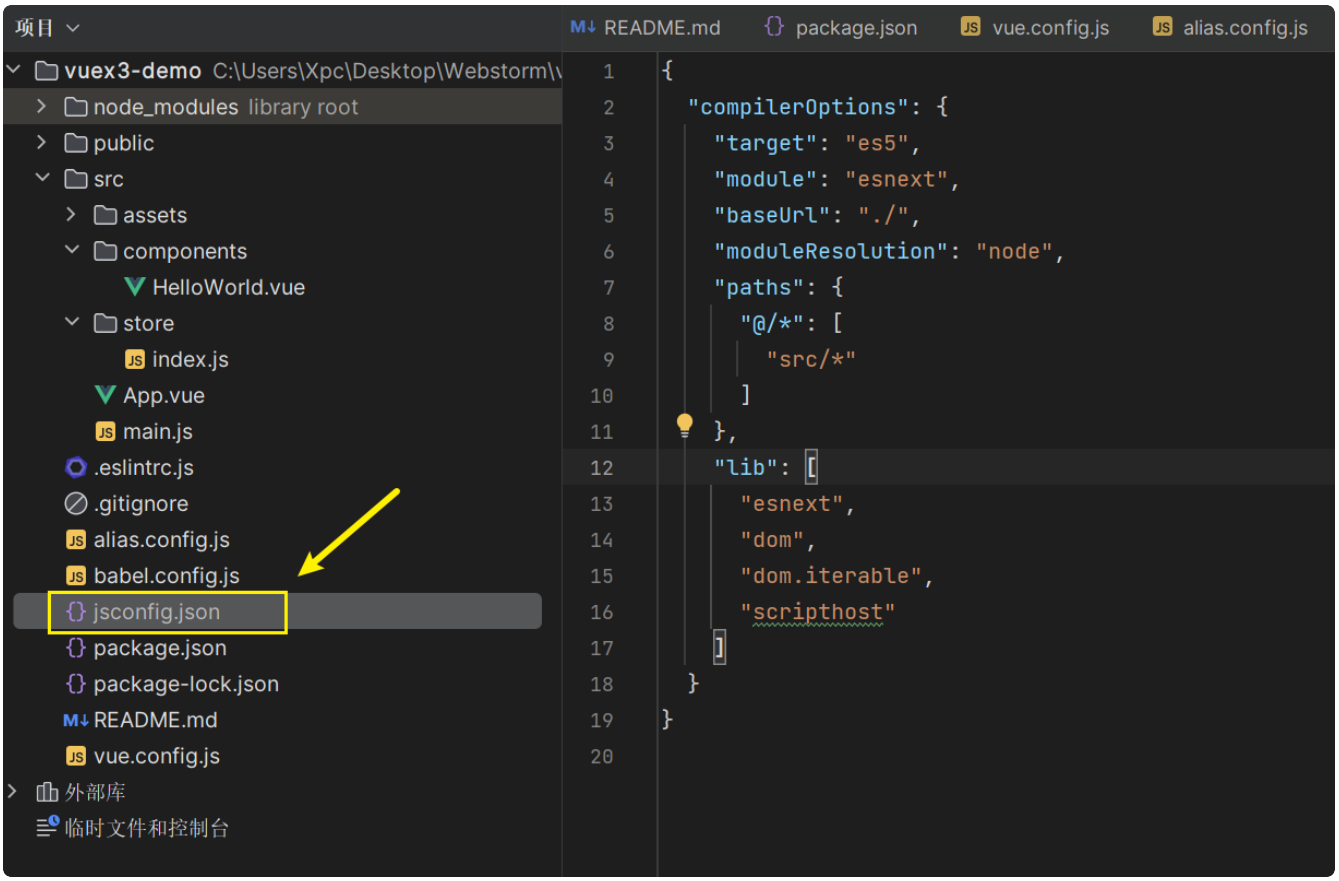
七、`@` 引起的 `WebStorm警告` 问题

alias.config.js JavaScript 复制代码

```
/**
 * tip: WebStorm解析使用
 *
 * @param dir
 * @returns {string}
 */
const resolve = dir => require('path').join(__dirname, dir);

module.exports = {
  resolve: {
    alias: {
      '@': resolve('src')
    }
  }
};
```



八、`jsconfig.json`这个文件在vue项目中有什么用?



> `jsconfig.json` 文件在 Vue 项目中的作用是配置 JavaScript 开发工具的一些行为和选项。（🤔 WebStorm无法解析，可能是给VSCode等其他IDE用的）
>
> `jsconfig.json` 是一个用于指定 JavaScript 项目根目录的配置文件，在 Vue 项目中它通常放置在项目的根目录下。它用于配置 JavaScript 开发工具（如编辑器、IDE、代码补全工具等）的一些行为和选项，以提供更好的开发体验和工具支持。
>
> 以下是一些常见的用例：
>
> 1. **智能代码补全**：`jsconfig.json` 可以配置你的 JavaScript 项目的基本设置，例如 `baseUrl`（基本路径），`paths`（文件别名），以及 `include` / `exclude`（指定哪些文件包括或排除在代码补全中）等，从而使编辑器或 IDE 能够更准确地提供智能代码补全建议。
> 2. **模块解析**：通过配置模块解析选项（如 `module`、`moduleResolution`、`resolveJsonModule` 等），`jsconfig.json` 可以影响 JavaScript 的模块解析行为，例如支持相对路径或绝对路径的模块导入、解析 `.json` 文件作为模块等。
> 3. **代码检查**：一些编辑器或 IDE 支持基于 `jsconfig.json` 进行代码检查和错误提示，可以根据该文件中配置的选项来验证 JavaScript 代码的正确性。
>
> 总之，`jsconfig.json` 文件用于为 JavaScript 项目提供工具支持和更好的开发体验。它配置了 JavaScript 开发工具的一些行为和选项，以提供代码补全、模块解析和代码检查等功能。在 Vue 项目中，这个文件可以帮助提高开发

> *效率和代码质量。*

END.