## ESLint安装

### Node.js安装

ESLint的运行环境需要Node.js(>=4.x)

Node.js v8.12.0 <https://nodejs.org/dist/v8.12.0/node-v8.12.0-x64.msi>

或者可以去官网下载更新的版本即可

### Npm简介

既然要用到Node.js,使用Node.js内置的npm安装ESLint也是需要;

庆幸Node.js安装之后,npm已经安装了;

Npm是一个Node.js环境下的依赖包管理器,用于安装运行在Node.js上的程序和插件;

常用命令npm install -g [ModuleName];

### 安装ESLint

ESLint的安装方式有2种,分全局安装和本地安装(项目内),根据实际使用,我们统一采用全局安装ESLint,运行以下命令行

//全局安装

# npm install -g eslint

\*ESLint的全局安装的路径其实是在Node.js指定的node\_module文件夹中;

### 同步配置文件

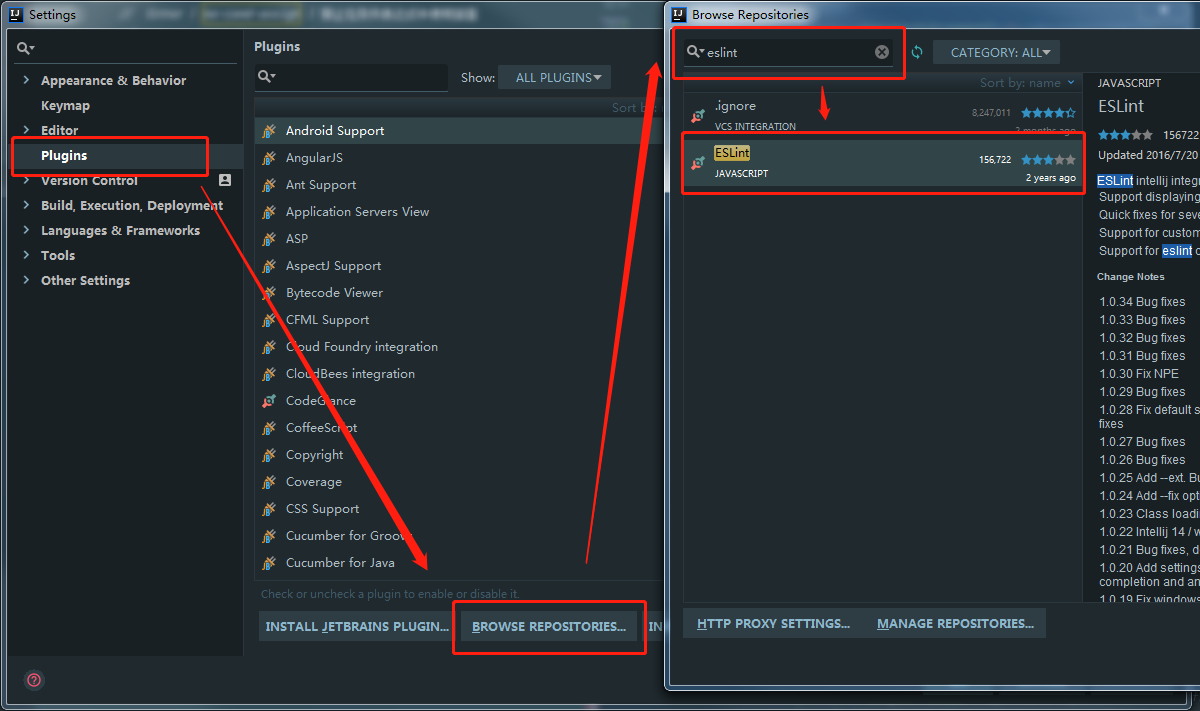
配置文件下载地址https://gitee.com/zicokuo/front\_end\_jseslint

点击链接可以下载zip压缩包,当然你也可以采用git clone

### ESLint插件安装与使用

#### IntelJ系列IDE

##### 安装

IntelJ 系列IDE Setting 配置

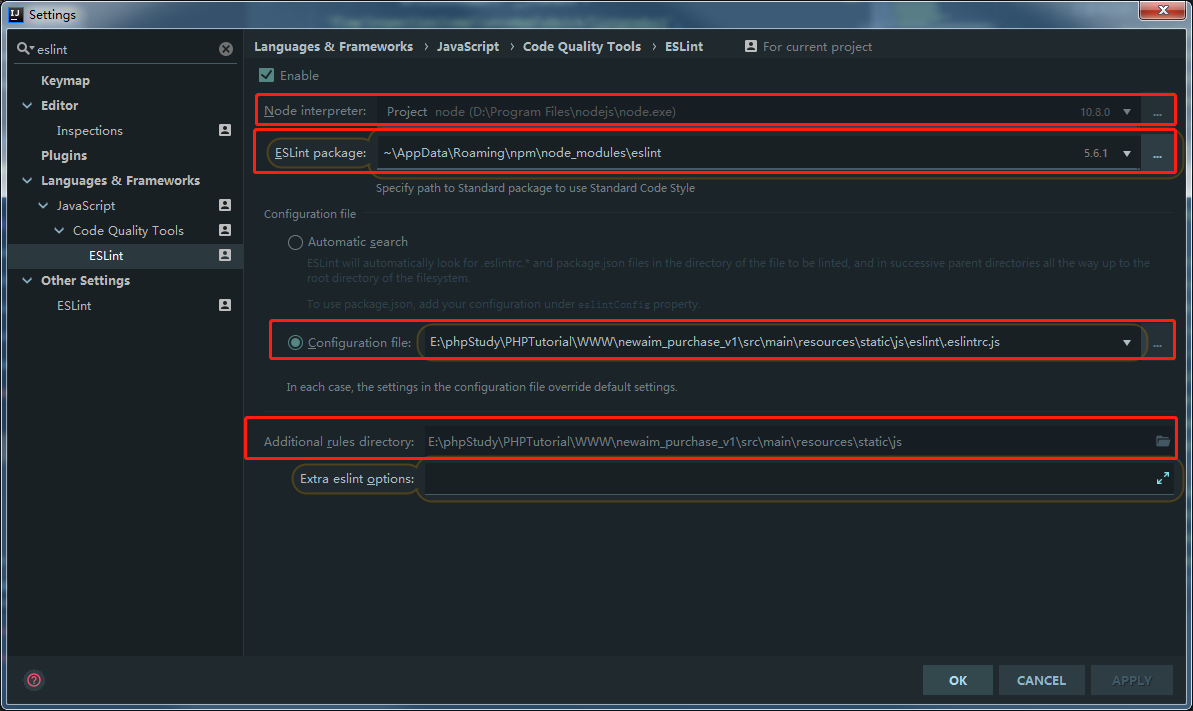
首先使用快捷键Ctrl + Alt + S打开IDE配置界面,也可以点击左上角菜单File->Settings;

继续点击Plugins->Browse Repositories...进入安装第三方插件;

输入eslint进行搜索,找到对应插件并安装和重启IDE即可;

##### 配置

IntelliJ 系列IDE使用插件需要进行配置,安装插件完毕之后,再次返回到Setting界面,搜索eslint即可进入配置界面



ESLint插件配置

###### Node interpreter

Node.js安装地址

一般插件会自动默认如果则手动修改

###### ESLint package

ESLint安装地址

全局安装的方式地址是~\AppData\Roaming\npm\node\_modules\eslint

###### Configuration file

项目中ESLint的配置文件

###### Additional rules directory

配置ESLint检索的根目录

一般为项目中需要检索的文件的根目录

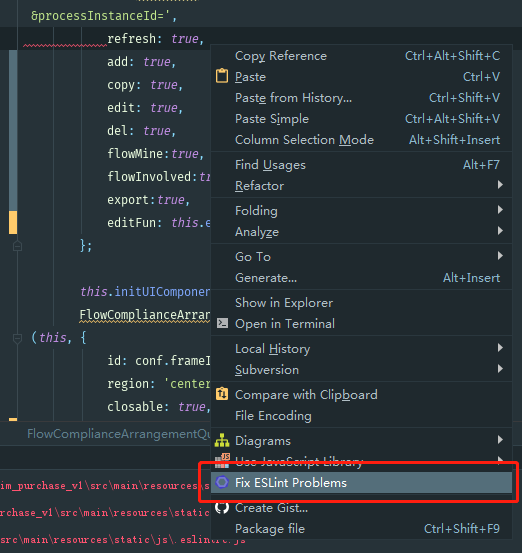
##### 使用

如上述步骤配置后,IDE中的ESLint已经起作用了,打开作用目录下的js文件可以看到新的规范提示;

##### 修复

ESLint中有着大部分的规则可以采用自动修复的方法进行处理

在当前文件夹右键菜单选择”Fix ESLint Problems”即可修复大部分问题;



IntelliJ IDE 自动修复

#### Sublime

##### 安装

Sublime使用ESLint需要安装2个插件SublimeLinter和SublimeLinter-eslint,使用控制台安装插件;

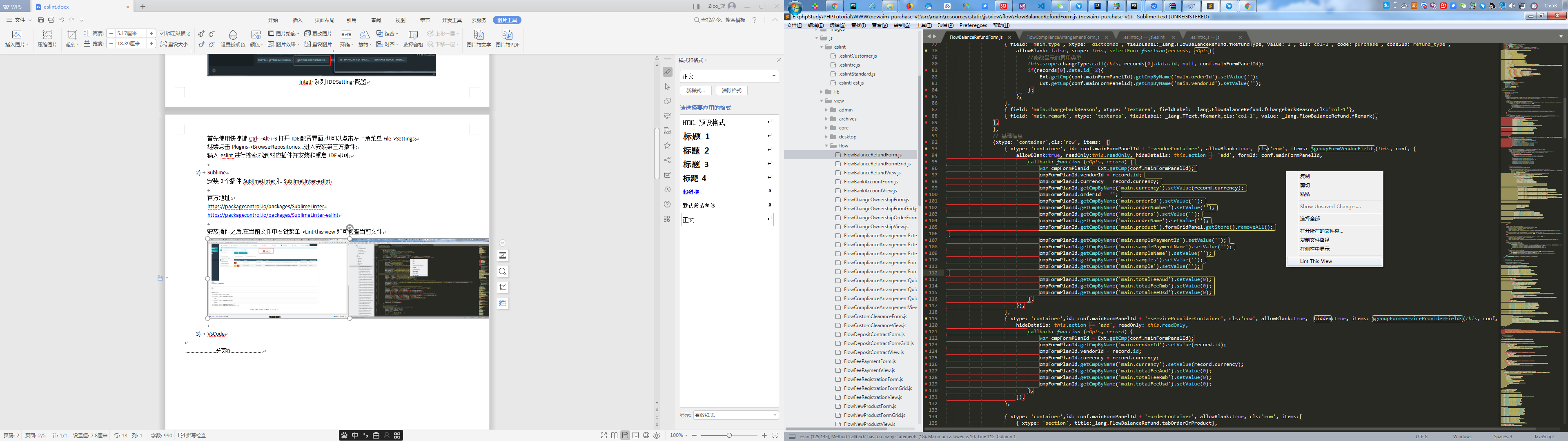
官方地址:

https://packagecontrol.io/packages/SublimeLinter

<https://packagecontrol.io/packages/SublimeLinter-eslint>

##### 使用

安装插件之后,在当前文件中右键菜单->Lint this view即可检查当前文件



##### 修复

Sublime的自动修复需要采用编译功能(build system)

* 1. 创建编译方法

点击工具(tool)->编辑系统(build system)->编译新系统(build new system)创建一个新的编译系统配置

输入以下代码段

{ "shell\_cmd": "eslint --fix $file" }

采用cmd运行eslint --fix 当前文件

然后保存到Sublime的Install Packagist文件夹中

* 1. 运行编译

再次点击工具(tool)->编译系统(build system)->eslint-fix进行编译当前文件,则会进行大部分规范的修复

### ESLint用例

*/\*\*  
 \* js ESLint 测试用例  
 \* 1.用例仅为常用用例,按照eslint/.eslintStandard.js作为标准编写  
 \* 2.本用例不涉及对Ext项目的配置  
 \* 3.本用例仅供参考  
 \** ***@author*** *Zico  
 \** ***@version*** *0.0.1  
 \*/  
  
var* example1 = *function* () {  
 *// Error | quotes | 引号类型强制统一采用单引号  
 var* testString1 = 'test';  
 *// Error | newline-after-var | 变量声明后是否需要空一行  
 var* testString2 = 'test';  
 testString2 += 1;  
 *// Error | operator-linebreak | 换行运算符要在行尾* testString2 = testString2  
 + 1;  
 *// Warning | semi | 对象结尾必须有分号*}  
  
*var* example2 = *function* () {  
 *var* testString1 = 'test';  
 *// Warning | quotes | 建议采用单引号  
 var* testString2 = "test";  
  
 *// Warning | no-implicit-coercion | 禁止隐式转换* testString1 = !!testString1;  
 *// Warning | vars-on-top | var建议放在作用域顶部  
 var* testSting3 = 'tes';  
};  
  
*var* example3 = *function* () {  
 *var* exp = *null*;  
 *try* {  
 *var* testString1='test1';  
 *var* testString2 ='test2';  
 *var* testString3= 'test3';  
 *var* testString4 = 'test4';  
 *// Warning | no-underscore-dangle | 标识符不能以\_开头  
 var* \_testString5 = 'test5';  
 *// Error | no-array-constructor | 禁止使用数组构造器  
 var* testArray1 = *new* Array(testString2, testString1, testString3,testString4,\_testString5);  
 *// Error | comma-dangle | 字面量项尾不能有逗号  
 var* testArray2 = [testString1, testString2, testString3,];  
 *// Error | no-sparse-arrays | 禁止稀疏数组  
 var* testArray3 = [testString1,,testString3];  
 *var* TEST\_OBJECT = *true*;  
 *// Error | camelcase | 没有驼峰命名  
 var* test\_object = {  
 \_private: *true*,  
 isCamelcase: TEST\_OBJECT,  
 *// Error | camelcase | 没有驼峰命名* check\_Camelcase: *function* (boolean) {  
 *return this*.isCamelcase;  
 *// Error | no-unreachable | 存在无法执行的代码  
 this*.isCamelcase = boolean;  
 },  
 };  
 *// Error | no-unneeded-ternary | 禁止不必要的嵌套  
 // Error | eqeqeq | 必须使用全等于 ===  
 var* testObject = test\_object.isCamelcase == test\_object.check\_Camelcase(*false*) ? *true* : *false*;  
 } *catch* (exp) {  
 *// Error | no-catch-shadow | 禁止catch子句与外部作用于变量同名  
 // Error | no-console | 禁止在生产环境使用console* console.log(exp);  
 }  
};  
  
*var* example4 = *function*(){  
 *var* testSuperFunction = *function*(){  
 *var* private1;  
 *var* private2;  
 *var* private3;  
 *var* private4;  
 *var* private5;  
 *var* private6;  
 *var* private7;  
 *var* private8;  
 *var* private9;  
 *var* private10;  
 *// Warning | max-statements | 函数内最多10个声明,大于10个时候应当抽离属性  
 var* private11;  
  
 *this*.status = 'stand';  
 };  
 *var* testChildFunction;  
  
 testSuperFunction.prototype = {  
 readyToGo:*function*(v){  
 *// Error | no-cond-assign | 禁止在条件表达式中使用赋值  
 if*(*this*.status = 'running'){  
 *return* 'running';  
 }*else*{  
 *return false*;  
 }  
 *return this*.go() + v;  
 },  
 *// Error | no-dupe-keys | 不允许使用重复的键名  
 // Error | no-dupe-args | 不允许重复的参数* readyToGo:*function* (v,v) {  
 *return* v;  
 }  
 };  
 *// Error | new-cap | 需要被new的function,请使用大写字母开头* testChildFunction = *new* testSuperFunction();  
  
 testChildFunction.sw = *function*(m){  
 *// Error | default-case | switch缺少default  
 switch* (m) {  
 *case* 1:  
 m = 1;  
 *// Warning | no-fallthrough | 建议不允许switch穿透,要break  
 case* 2:  
 *break*;  
 *// Error | no-duplicate-case | 不允许重复的case  
 case* 1:  
 *return*;  
 }  
 };  
  
};