# 

[**命名原則\_Naming Rules**](#_fripsc3218rc) **3**

[檔案命名\_File names](#_2qb3nmg9377j) 3

[組件命名\_Component names](#_c6q9502eyx36) 3

[組件在模版中的命名規則\_Component name casing in templates](#_ivwa319te5oh) 3

[基礎組件命名\_Base component names](#_z0tj1egyeo31) 4

[單例組件名\_Single-instance component names](#_wtg5rzaj628n) 4

[緊密耦合的組件\_Tightly coupled component names](#_fkxf3e3ohxyu) 4

[組件命名順序性\_Order of words in component names](#_mmitwsy09nfs) 4

[類別命名\_Class names](#_fja0frme5l98) 4

[方法命名\_Method names](#_bzs0yga96rh0) 4

[常數命名\_Constant names](#_ijinrunazrk1) 4

[全域變數命名\_Global variable names](#_ff8r9pip8wec) 4

[區域變數命名\_Local variable names](#_su4bmkoh70e) 4

[參數命名\_Parameter names](#_p7mj9apqkcq3) 5

[Props 命名\_Props names](#_m3pagpv53iap) 5

[私有屬性$\_ 做為前綴命名\_Private property names](#_no62t1aq6wl9) 5

[使用prototype應用 $ 做為前綴命名\_Prototype property names](#_gymm5a2sjsmd) 5

[**寫作規範\_Coding guideline**](#_xcrcfg5fr399) **6**

[組件數據必須是函數](#_z4x1fn8tytr9) 6

[Props 定義應該盡量詳細](#_wdag4qzavmsy) 6

[總是用 key 配合 v-for，避免v-if和v-for用在一起](#_43akgld6vide) 7

[模板（template）每個特性單獨一行](#_kdrmyl4mj6b3) 7

[模板（template）中簡單的表達式](#_b1t0f58e5i2s) 7

[模板（template）中指令縮寫](#_su11jolgdbf3) 8

[模板（template）中attribute 值帶引號](#_acg029volyf3) 8

[模板（template）中盡可能在v-if/v-if-else/v-else使用key](#_7euoy6u9q7gf) 8

[**其他規範\_Other Rules**](#_qqs5anubcj91) **9**

[JS強型別檢查](#_4z8qnxaz5058) 9

[if/for的使用](#_hkugqjpqsd57) 9

[應避免使用多層三元運算子](#_n5p3vdcn7yk9) 9

[變數宣告](#_2vejy75jyl2a) 9

[使用import 取代require，使用export取代module.exports](#_8yqubsknvzkj) 9

[不要在模塊輸入使用通配符 \*](#_4ubpxnfwnnas) 10

[Arrow function 更勝過 Regular function](#_ynmr7nbvu9t7) 10

[程式上版前製作業](#_xb4mr3xk2w65) 10

[Package.json裡套件的依賴應該要精準地被使用](#_gdt8dshlnxbk) 10

[非正規化寫法，雖然可以編譯成功，但以團隊開發角度應該避免](#_u1w89zkgjepi) 10

[避免hard code、dead code](#_jayps8j43uz0) 11

[**程式碼結構\_Source File Structure**](#_bhub1ied177k) **11**

[結構順序](#_y9n86ciayj1t) 11

[**註解\_Annotation**](#_b0nqe2bxurph) **11**

[時機及規則](#_7q8pyk3eyh3w) 11

[Function註解格式](#_dwpqj08z7k4c) 12

[**測試\_Testing**](#_p8mp3x60633v) **12**

[先寫測試再寫程式](#_xe31plyo560w) 12

[先寫程式再寫測試](#_ip0lmw5l1gad) 12

[如何容易測試](#_s76t261exswe) 12

[**格式\_Formatting**](#_egxv549urs65) **13**

[VS code透過ESLint 來強制規範寫作風格](#_lc3hxtc1vgb0) 13

[VS code透過IDE來協助你的排版風格](#_qmjs88en6erj) 14

[WebStorm透過ESLint 來強制規範寫作風格](#_m2jjrl9k5nfu) 14

[**參考\_Reference**](#_dkwaajy8h2y8) **15**

[Google Javascript Style Guide](#_cuxr4witijw4) 16

[Airbnb Javascript Style Guide](#_w8e7w8xsxw7u) 16

[JavaScript Standard Style Guide](#_uymqplscsmxz) 16

# 命名原則\_Naming Rules

### 檔案命名\_File names

* 1. 大駝峰（PascalCase），無底線，不能縮寫
  2. 橫線連接（kebab-case），無底線，不能縮寫  
     Ex. TodoList.vue、todo-list.vue

### 組件命名\_Component names

* 1. 大駝峰（PascalCase）
  2. 全局組件註冊時，可以兼容使用橫線連接(kebab-case)
  3. 組件名必須為多個單詞

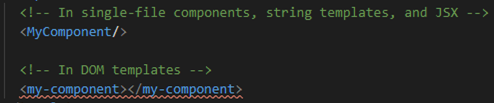
Ex.



### 組件在模版中的命名規則\_Component name casing in templates

* 1. 在single file component、string templates、JSX使用大駝峰（PascalCase）
  2. 在DOM templates 使用橫線連接(kebab-case)
  3. 或者全部使用橫線連接(kebab-case)

Ex.



### 基礎組件命名\_Base component names

* 1. 應用特定樣式和約定的基礎組件(也就是展示類的、無邏輯的或無狀態的組件)應該全部以一個特定的前綴開頭，比如Base、App或V

Ex. BaseButton.vue

### 單例組件名\_Single-instance component names

* 1. 對於每個頁面只會使用一次的組件只應該擁有單個活躍實例的組件，應該以 The 前綴命名，以示其唯一性，這些組件永遠不接受任何 props

Ex. TheHeader.vue

### 緊密耦合的組件\_Tightly coupled component names

* 1. 和父組件緊密耦合的子組件應該以父組件名作為前綴命名

Ex. components/

| - Searchbar.vue

| - SearchbarNavigation.vue

### 組件命名順序性\_Order of words in component names

* 1. 組件名應該以層次較高、較普遍的單詞開頭，以描述性的修飾詞結尾

Ex. components/

| - SearchButtonClear.vue

| - SearchButtonRun.vue

| - SearchInputQuery.vue

### 類別命名\_Class names

* 1. 大駝峰（PascalCase）  
     Ex.export class UserInfo { ... }

### 方法命名\_Method names

* 1. 小駝峰(camelCase)  
     Ex. sendMessage () { ... }

### 常數命名\_Constant names

* 1. 常數變數小駝峰(camelCase)

Ex. const treeTable = ‘text’

* 1. 全大寫，單字間以底線連接，可以被兼容

Ex. const VILLAINS\_URL = ‘api/villains’

### 全域變數命名\_Global variable names

* 1. 小駝峰(camelCase)  
     Ex. this.computedValues

### 區域變數命名\_Local variable names

* 1. 小駝峰(camelCase)

Ex. let selectedItem

### 參數命名\_Parameter names

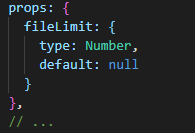
* 1. 小駝峰(camelCase)

Ex. onCreate ( savedState) { ... }

### Props 命名\_Props names

* 1. 在聲明 props 的時候，其命名應該始終使用小駝峰(camelCase)
  2. 在模板（template）和 JSX 中應該始終使用橫線連接 (kebab-case)

Ex.





### 私有屬性$\_ 做為前綴命名\_Private property names

* 1. 在 plugin、mixin 中的私有屬性始終使用 $\_ 做為前綴，同時，為了迴避和其它作者的衝突，再附帶一個命名空間(小駝峰)以迴避和其它作者的衝突

Ex. $\_myNameSpace\_updateData ()

### 使用prototype應用 $ 做為前綴命名\_Prototype property names

* 1. 官方建議 prototype 的命名最好冠上 $ 符號，原因是避免與組件內的方法名稱衝到，另外就是開發慣例(ex. this.$router, this.$store) 好讓所有使用者一眼就可以理解這就是 prototype ！

Ex. this.$router ,this.$store

# 寫作規範\_Coding guideline

### 組件數據必須是函數

* 1. 當在組件中使用data property的時候(除了new Vue外的任何地方)，它的值必須是返回一個對象的函數。

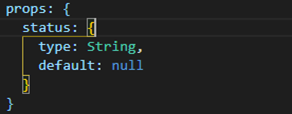
Ex.



### Props 定義應該盡量詳細

* 1. props 的定義應該盡量詳細，至少需要指定其類型，與其默認值。

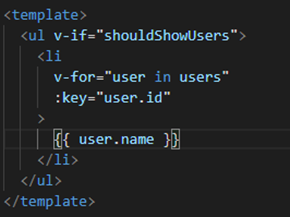
Ex.



### 

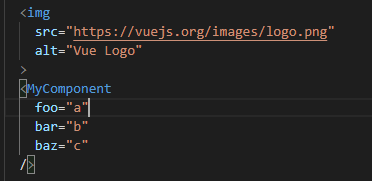
### 總是用 key 配合 v-for，避免v-if和v-for用在一起

Ex.



### 模板（template）每個特性單獨一行

Ex.



### 模板（template）中簡單的表達式

* 1. 組件模板應該只包含簡單的表達式，複雜的表達式則應該重構為計算屬性或方法

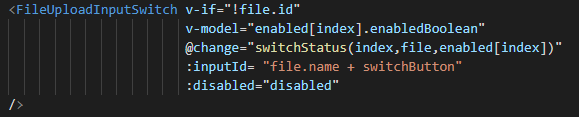
Ex.



### 模板（template）中指令縮寫

* 1. 用 : 表示v-bind:，用 @ 表示v-on:和用 # 表示v-slot:

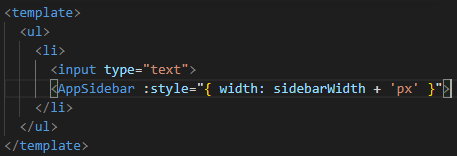
Ex.



### 模板（template）中attribute 值帶引號

* 1. 在HTML中不帶空格的attribute值是可以沒有引號的，但是這種做法通常會導致因避免使用空格，從而使屬性值的可讀性降低。

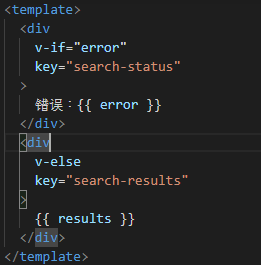
Ex.



### 模板（template）中盡可能在v-if/v-if-else/v-else使用key

* 1. 如果一組v-if+ v-else的元素類型相同，最好使用key

Ex.



# 其他規範\_Other Rules

### JS強型別檢查

在Javascript判斷上，請使用=== / !==，加強型別檢查。

### if/for的使用

* 1. 在單一行的if / for，雖然可以允許不括號，但也請寫 { }。
  2. 應避免If /else If /else 過多層，造成巢狀結構 。

### 應避免使用多層三元運算子

多層三元運算式不容易維護，更容易造成程式碼過長，不易維護。

Ex. A判斷式 ? A條件成立 : A不成立使用B判斷式 ? B條件成立 : B不成立

### 變數宣告

因為var 是function scope、而let是block scope、const是常數不能被修改，var 使用上會有循環變數洩漏全局變數問題與內區塊變數覆蓋外區塊變數問題

Global →Function → Block，精準的使用let、const，取代使用var



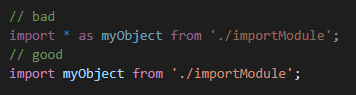
### 使用import 取代require，使用export取代module.exports

[在ES6之後JS有了模塊的概念](https://zhuanlan.zhihu.com/p/33843378)

### 

### 不要在模塊輸入使用通配符 \*

我們應該要更精確使用輸入，而不是直接輸入整體



### Arrow function 更勝過 Regular function

我們常在使用Regular function時沒辦法使用this指向變數或函數

請使用arrow function ()=>，避免用let OOO = function (){ }; OOO.bind(this);

### 程式上版前製作業

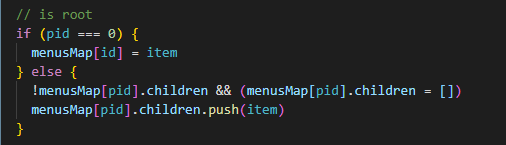
先拿掉開發時之程式碼、console.log、debugger，或用不到之註解、註解程式碼

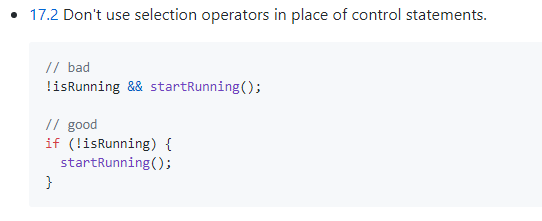
### Package.json裡套件的依賴應該要精準地被使用

Dependencies是程式上線的依賴，其餘放到devDependencies開發階段的依賴

### 非正規化寫法，雖然可以編譯成功，但以團隊開發角度應該避免

Ex.

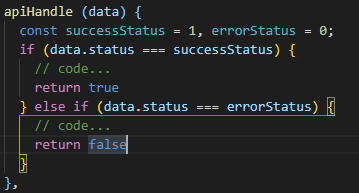
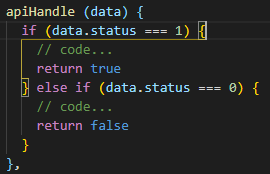




[Airbnb JavaScript Style Guide](https://github.com/airbnb/javascript)

### 避免hard code、dead code

Ex.



# 程式碼結構\_Source File Structure

### 結構順序

* 1. template(HTML)
  2. script(JS)
  3. style(CSS)

<template>...</template>

<script>/\* ... \*/</script>

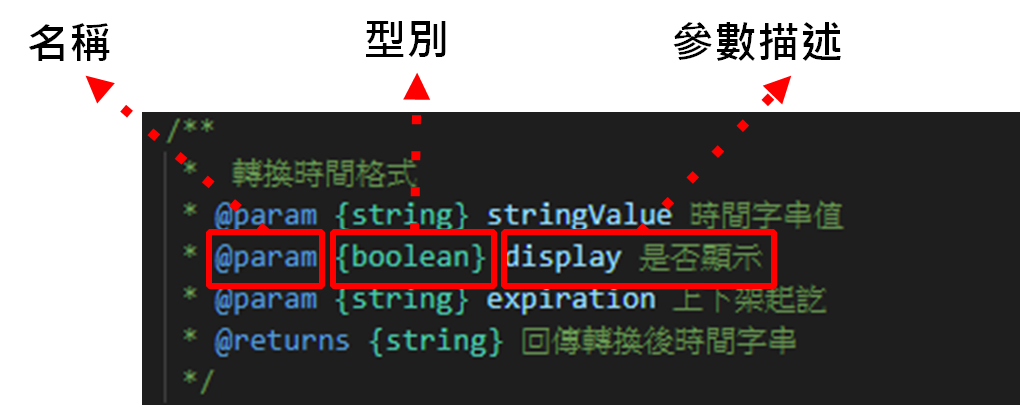
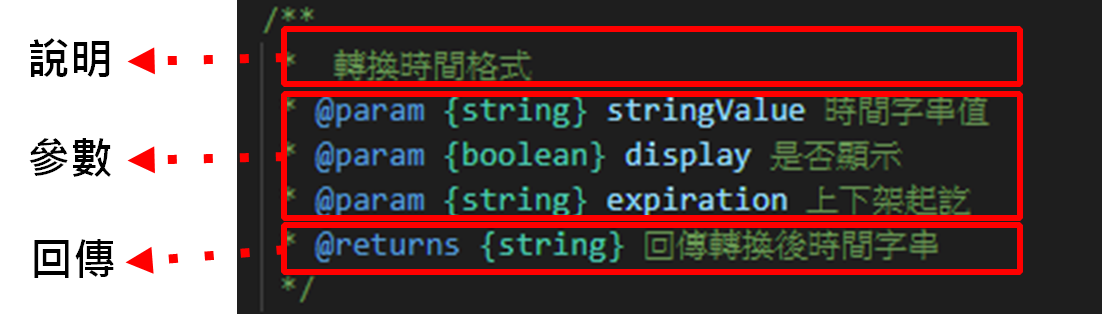
<style>/\* ... \*/</style>

# 註解\_Annotation

### 時機及規則

* 1. 以利於快速了解複雜程式碼為原則，若透過命名可快速理解功能，則不需加註解。
  2. 特殊邏輯、不易理解，必加註解說明
  3. 暫時用的程式碼必須加上 //TODO，並寫上原因  
     後續可使用搜尋的TODO視窗快速找出這些待調整的程式碼
  4. 註解請寫在程式碼上方，避免寫在右方，容易因過長而不易閱讀

### Function註解格式



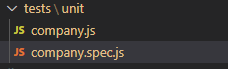
# 測試\_Testing

### 先寫測試再寫程式

* 1. 依據TDD測試五步驟

寫測試 => 紅燈(fail) => 寫程式 => 綠燈(success) =>重構(寫Vue)

Ex.在test/unit底下新增OOO.spec.js、OOO.js



### 先寫程式再寫測試

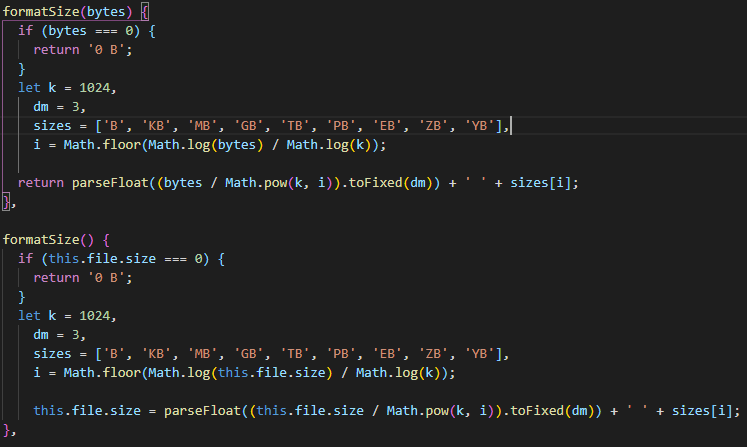
* 1. 思考使用情境 => 寫程式 => 寫測試

Ex.在OOO.vue 同一層新增 OOO.test.js



### 如何容易測試

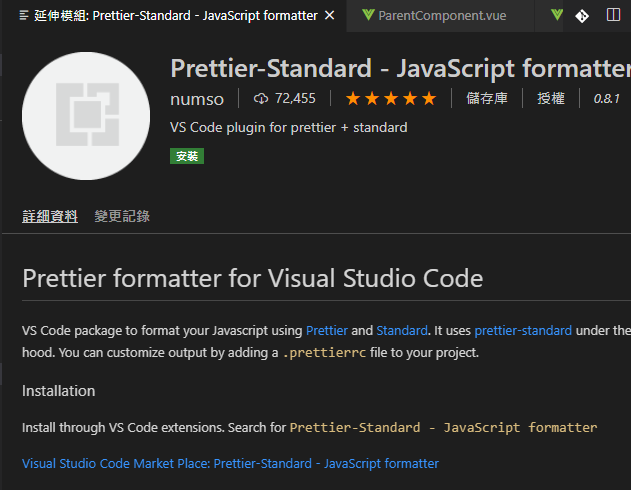
* 1. 善用區域變數、參數與回傳值，讓你的函數好測試



# 格式\_Formatting

### VS code透過ESLint 來強制規範寫作風格

* 1. 使用vue/standard風格規範開發習慣



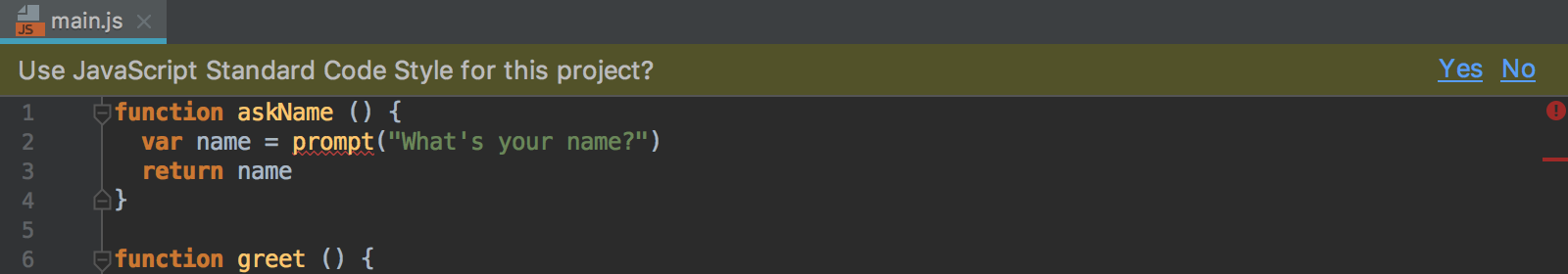
安裝Prettier-Standard - JavaScript formatter 延伸模組

### VS code透過IDE來協助你的排版風格

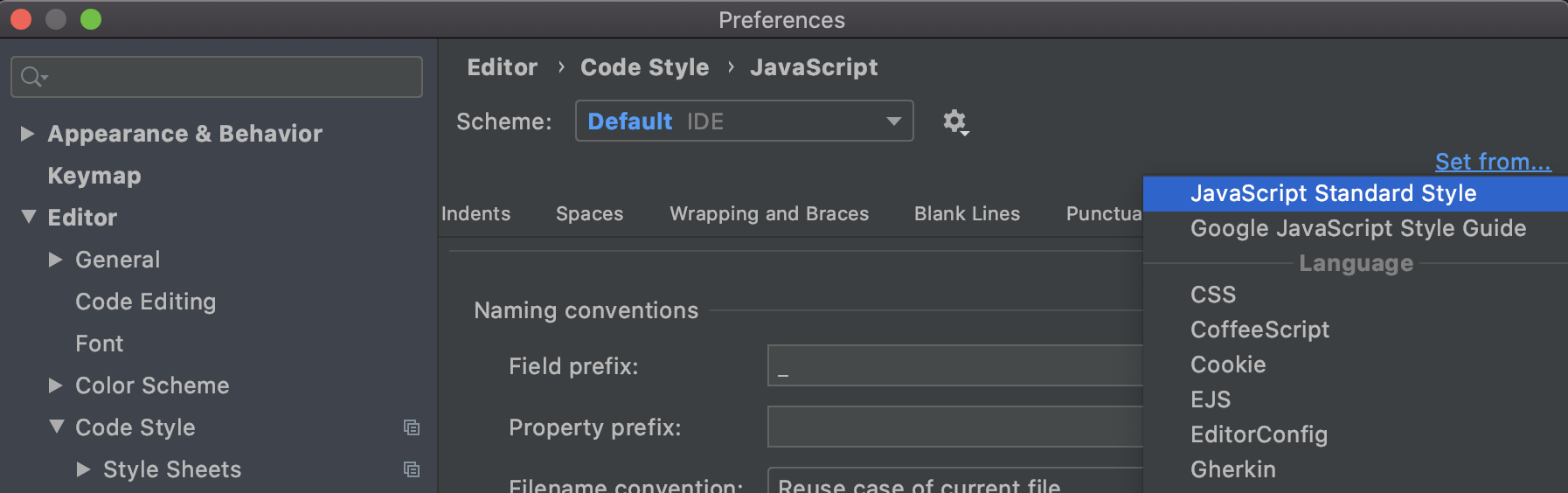
* 1. shift+alt+F 快速調整縮排

### WebStorm透過ESLint 來強制規範寫作風格

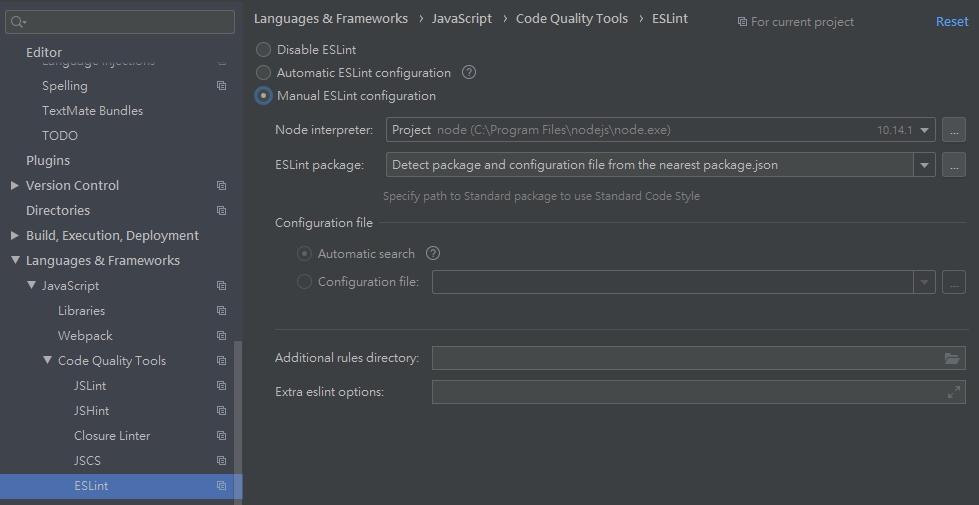
只要你在初始化專案時選用ESLint + Standard config，webstorm就會自動地幫你使用Standard 的建議，也有機會出現以下選項，按YES



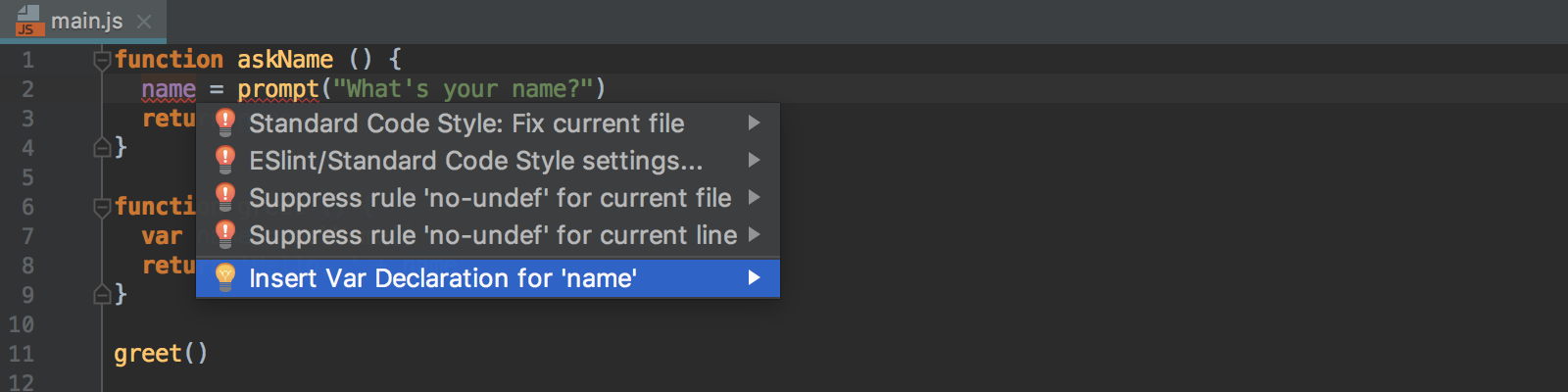
若要手動更改，可以到Preferences/Settings | Editor | Code Style | JavaScript, 按Set from...設定(公司規定是用Standard Style)



若要選用自己的ESLint Standard 風格，可以到Preferences/Settings | Languages & Frameworks | JavaScript | Code Quality Tools | ESLint, 選擇 Manual ESLint configuration，並且選用檔案的起始位置(如果有需客製化風格可以向團隊提出來)



透過快捷鍵alt+enter，快速修正程式碼



# 參考\_Reference

### Google Javascript Style Guide

<https://google.github.io/styleguide/jsguide.html>

### Airbnb Javascript Style Guide

<https://github.com/airbnb/javascript>

### JavaScript Standard Style Guide

<https://standardjs.com/rules-zhtw.html>