### **jshint配置**

###### **三种配置方式**

* 通过 --config标记手动配置;
* 使用 ****.jshintrc**** 文件:

这个设置允许您每个项目有不同的配置文件。把你的文件放到项目根目录,只要你运行JSHint从任何地方在你的项目目录树,将使用相同的配置文件。

* 配置放到项目的 ****package.json**** 文件里面， ****jshintConfig**** 下面  
  配置文件是一 个简单的JSON文件,指定JSHint选项来打开或关闭。例如,以下文件将启用警告未定义的和未使用的变量和告诉JSHint全局变量命名 MY\_GLOBAL

{

"undef": true,

"unused": true,

"predef": [ "MY\_GLOBAL" ]

}

#### **内联配置**

* 除了使用配置文件,您可以在你的文件中使用jshint或者globals开头，并跟着配置项以冒号分隔值列表来配置JSHint。例如,下面的代码片段将启用警告未定义的和未使用的变量和告诉JSHint全局变量命名MY\_GLOBAL。

/\* jshint undef: true, unused: true \*/

/\* globals MY\_GLOBAL \*/

* 可以使用单行或者多行来配置JSHint，如果放在函数里面，则只影响该函数。

### **JSHint 忽略**

* ignore 告诉jshint忽略一个代码块

// Code here will be linted with JSHint.

/\* jshint ignore:start \*/

// Code here will be ignored by JSHint.

/\* jshint ignore:end \*/

* 在jshint ****ignore:start****和 ****ignore:end****之间的所有代码都将被JSHint忽略,所以你可以使用像[Facebook React](http://facebook.github.io/react/" \t "http://www.jianshu.com/p/_blank)这样的扩展框架。
* 可以忽略一行:

ignoreThis(); // jshint ignore:line

### **JSHint配置选项**

* [bitwise](http://jshint.com/docs/options/" \l "bitwise" \t "http://www.jianshu.com/p/_blank)  
  禁用位运算符，位运算符在 JavaScript 中使用较少，经常是把 && 错输成 &。

bitwise: true

* [camelcase](http://jshint.com/docs/options/" \l "camelcase" \t "http://www.jianshu.com/p/_blank)  
  警告:此选项已被弃用,将在JSHint的下一个主要版本被删除.  
  此选项可以强制所有变量名称为使用驼峰风格或UPPER\_CASE用下划线。

camelcase:true/false

* [curly](http://jshint.com/docs/options/" \l "curly" \t "http://www.jianshu.com/p/_blank)  
  循环或者条件语句必须使用花括号包围.  
  这个选项需要你总是把花括号在块循环和条件。JavaScript块时可以省略括号包含只有一个语句,例如:

while (day)

shuffle();

* 然而,在某些情况下,它会导致错误(你可能会认为 sleep()是一个循环的一部分,而事实上它不是)

while (day)

shuffle();

sleep();

* [enforceall](http://jshint.com/docs/options/" \l "enforceall" \t "http://www.jianshu.com/p/_blank)  
  警告:此选项已被弃用,将在JSHint的下一个主要版本被删除.  
  它启用所有强制执行选项和禁用该版本中定义的所有的[Relaxing options](http://jshint.com/docs/options/" \l "relaxing-options" \t "http://www.jianshu.com/p/_blank)；
* [eqeqeq](http://jshint.com/docs/options/" \l "eqeqeq" \t "http://www.jianshu.com/p/_blank)  
  设置为true,禁止使用这个选项 ==和 !=，强制使用 ===和 !==。

eqeqeq: true

* [es3](http://jshint.com/docs/options/" \l "es3" \t "http://www.jianshu.com/p/_blank)  
  警告:此选项将在JSHint的下一个主要版本被删除,使用esversion: 3代替.  
  使用ECMAScript 3规范。使用这个选项主要为了兼容低级浏览器 IE 6/7/8/9-and其他遗留JavaScript环境。
* [es5](http://jshint.com/docs/options/" \l "es5" \t "http://www.jianshu.com/p/_blank)  
  警告:此选项将被删除在JSHint的下一个主要版本，使用 esversion: 5代替。  
  这个选项允许语法中定义ECMAScript 5.1规范，这包括允许保留关键字作为对象属性。
* [esversion](http://jshint.com/docs/options/" \l "esversion" \t "http://www.jianshu.com/p/_blank)  
  这个选项用于指定的ECMAScript版本代码必须遵循。它可以假设以下值之一:  
  3--如果你需要可执行程序在老这类浏览器Internet Explorer 6/7/8/9-and其他遗留JavaScript环境  
  5--先使语法中定义ECMAScript 5.1规范。这包括允许保留关键字作为对象属性。  
  6--告诉JSHint代码使用ECMAScript 6具体的语法。请注意,并不是所有的浏览器都实现它们。
* [forin](http://jshint.com/docs/options/" \l "forin" \t "http://www.jianshu.com/p/_blank)  
  这个选项要求所有 for in循环过滤对象的item。他在声明中允许for遍历一个对象所有属性的名称包括通过原型链继承来的属性。

for (key in obj) {

if (obj.hasOwnProperty(key)) {

// We are sure that obj[key] belongs to the object and was not inherited.

}

}

* [freeze](http://jshint.com/docs/options/" \l "freeze" \t "http://www.jianshu.com/p/_blank)  
  这个选项禁止重写原生对象的原型列如 Array, Date等等。
* [funcscope](http://jshint.com/docs/options/" \l "funcscope" \t "http://www.jianshu.com/p/_blank)  
  禁止从外部访问内部声明的变量

function test() {

if (true) {

var x = 0;

}

x += 1; // Default: 'x' used out of scope.

// No warning when funcscope:true

}

* [futurehostile](http://jshint.com/docs/options/" \l "futurehostile" \t "http://www.jianshu.com/p/_blank)  
  允许警告js未来版本中定义的标识符。
* [globals](http://jshint.com/docs/options/" \l "globals" \t "http://www.jianshu.com/p/_blank)  
  这个选项可以用来指定一个没有正式定义的全局变量的白名单。  
  配置 globals在单个文件,看看内联配置.
* [immed](http://jshint.com/docs/options/" \l "immed" \t "http://www.jianshu.com/p/_blank)  
  警告:此选项已被弃用,将在JSHint的下一个主要版本被删除。  
  需要直接调用的函数必须被括号包围，如：（function（）{}（））;
* [indent](http://jshint.com/docs/options/" \l "indent" \t "http://www.jianshu.com/p/_blank)  
  警告:此选项将在JSHint的下一个主要版本被删除。  
  设置代码缩进长度
* [iterator](http://jshint.com/docs/options/" \l "iterator" \t "http://www.jianshu.com/p/_blank)  
  禁止****iterator****属性有关的警告。  
  此属性不支持所有浏览器所以小心使用它。
* [latedef](http://jshint.com/docs/options/" \l "latedef" \t "http://www.jianshu.com/p/_blank)  
  禁止定义之前使用变量。  
  这个选项设置为“nofunc”将允许函数声明被忽略。
* [maxcomplexity](http://jshint.com/docs/options/" \l "maxcomplexity" \t "http://www.jianshu.com/p/_blank)  
  设置代码文件独立直线路径最大复杂度检测。
* [maxdepth](http://jshint.com/docs/options/" \l "maxdepth" \t "http://www.jianshu.com/p/_blank)  
  设置代码最大嵌套深度。

// jshint maxdepth:2

function main(meaning) {

var day = true;

if (meaning === 42) {

while (day) {

shuffle();

if (tired) {

// JSHint: Blocks are nested too deeply (3).

sleep();

}

}

}

}

* [maxerr](http://jshint.com/docs/options/" \l "maxerr" \t "http://www.jianshu.com/p/_blank)  
  设置JSHint最大警告数。默认50
* [maxlen](http://jshint.com/docs/options/" \l "maxlen" \t "http://www.jianshu.com/p/_blank)  
  警告:此选项将在JSHint的下一个主要版本被删除  
  设置最大行数

[maxparams](http://jshint.com/docs/options/" \l "maxparams" \t "http://www.jianshu.com/p/_blank)  
这个选项允许您设置每个函数的形参最大数量

// jshint maxparams:3

function login(request, onSuccess) {

// ...

}

// JSHint: Too many parameters per function (4).

function logout(request, isManual, whereAmI, onSuccess) {

// ...

}

* [maxstatements](http://jshint.com/docs/options/" \l "maxstatements" \t "http://www.jianshu.com/p/_blank)  
  这个选项允许您设置语句允许的最大声明数:  
  // shint maxstatements:4

function main() {

var i = 0;

var j = 0;

// Function declarations count as one statement. Their bodies

// don't get taken into account for the outer function.

function inner() {

var i2 = 1;

var j2 = 1;

return i2 + j2;

}

j = i + j;

return j; // JSHint: Too many statements per function. (5)

}

* 函数也算声明。
* [newcap](http://jshint.com/docs/options/" \l "newcap" \t "http://www.jianshu.com/p/_blank)  
  警告此选项已被弃用,将被删除在JSHint的下一个主要版本  
  要求所有构造器使用new F()形式。
* [noarg](http://jshint.com/docs/options/" \l "noarg" \t "http://www.jianshu.com/p/_blank)  
  禁止使用这个选项 arguments.caller和 arguments.callee。这两个 .caller  
  和.callee将会被弃用。事实上,ECMAScript 5 严格模式禁止使用arguments.callee
* [nocomma](http://jshint.com/docs/options/" \l "nocomma" \t "http://www.jianshu.com/p/_blank)  
  这个选项禁止使用逗号操作符。
* [noempty](http://jshint.com/docs/options/" \l "noempty" \t "http://www.jianshu.com/p/_blank)  
  警告此选项已被弃用,将被删除在JSHint的下一个主要版本。  
  空代码块警告。
* [nonbsp](http://jshint.com/docs/options/" \l "nonbsp" \t "http://www.jianshu.com/p/_blank)  
  不换行的空格警告
* [nonew](http://jshint.com/docs/options/" \l "nonew" \t "http://www.jianshu.com/p/_blank)  
  这个选项禁止使用new构造器函数。有些人喜欢调用构造函数，但是不赋值给任何对象:

new MyConstructor();

* [notypeof](http://jshint.com/docs/options/" \l "notypeof" \t "http://www.jianshu.com/p/_blank)  
  检查无效 typeof操作符的值

// 'fuction' instead of 'function'

if (typeof a == "fuction") { // Invalid typeof value 'fuction'

// ...

}

* [predef](http://jshint.com/docs/options/" \l "predef" \t "http://www.jianshu.com/p/_blank)  
  扩展的隐式全局变量
* [quotmark](http://jshint.com/docs/options/" \l "quotmark" \t "http://www.jianshu.com/p/_blank)  
  警告此选项已被弃用,将被JSHint的下一个主要版本删除。  
  这个选项执行代码中使用引号的一致性。它接受三个值:
  + true-- 代码字符串禁止单引号双引号混用,
  + "single"--只允许单引号
  + "double"--只允许双引号。
* [shadow](http://jshint.com/docs/options/" \l "shadow" \t "http://www.jianshu.com/p/_blank)  
  检查变量重复定义  
  他接受4个值：
  + "inner" 只检查是否在相同的作用域重复定义
  + "outer" 检查外部作用域
  + false 与inne一样
  + true 允许变量覆盖
* [singleGroups](http://jshint.com/docs/options/" \l "singleGroups" \t "http://www.jianshu.com/p/_blank)  
  禁止使用分组操作符

// jshint singleGroups: true

delete(obj.attr); // Warning: Unnecessary grouping operator.

* [strict](http://jshint.com/docs/options/" \l "strict" \t "http://www.jianshu.com/p/_blank)  
  ECMAScript 5严格模式
  + "global" - 全局层面的严格模式"use strict";
  + "implied" - 文件里面使用"use strict";
  + false - 禁止使用严格模式
  + true - 函数上面必须使用一个"use strict";
* [undef](http://jshint.com/docs/options/" \l "undef" \t "http://www.jianshu.com/p/_blank)  
  变量未定义

// jshint undef:truefunction test() {

var myVar = 'Hello, World';

// Oops, typoed here. JSHint with undef will complain

console.log(myvar);

}

如果你的另一个文件中定义的变量,你可以使用 global指令告诉JSHint。

* [unused](http://jshint.com/docs/options/" \l "unused" \t "http://www.jianshu.com/p/_blank)  
  变量定义未使用

// jshint unused:true

function test(a, b) {

var c, d = 2;

return a + d;

}

test(1, 2);

// Line 3: 'b' was defined but never used.

// Line 4: 'c' was defined but never used

* [varstmt](http://jshint.com/docs/options/" \l "varstmt" \t "http://www.jianshu.com/p/_blank)  
  设置为true时，禁止使用var声明变量

// jshint varstmt: true

var a; // Warning: `var` declarations are forbidden. Use `let` or `const` instead.

### **宽松选项**

* 设置为true时,这些选项会使代码JSHint产生更少的警告。
* [asi](http://jshint.com/docs/options/" \l "asi" \t "http://www.jianshu.com/p/_blank)  
  禁止缺少分号警告
* [boss](http://jshint.com/docs/options/" \l "boss" \t "http://www.jianshu.com/p/_blank)  
  禁止比较表达式的值没有达到预期警告。  
  通常情况下,代码 if (a = 10) {}是一个错误,但他有可能这样用

for (var i = 0, person; person = people[i]; i++) {}

* 你可以禁止这个错误，比如

for (var i = 0, person; (person = people[i]); i++) {}

* [debug](http://jshint.com/docs/options/" \l "debug" \t "http://www.jianshu.com/p/_blank)  
  忽略 debugger
* [elision](http://jshint.com/docs/options/" \l "elision" \t "http://www.jianshu.com/p/_blank)  
  告诉JSHint代码使用ES3数组省略元素,或空元素(例如, [1, , , 4, , , 7]  
  ).
* [eqnull](http://jshint.com/docs/options/" \l "eqnull" \t "http://www.jianshu.com/p/_blank)  
  禁止 == null比较。通常这样的比较有用,当你想检查一个变量是否null  
  或 undefined
* [eqnull](http://jshint.com/docs/options/" \l "eqnull" \t "http://www.jianshu.com/p/_blank)  
  警告此选项将在JSHint的下一个主要版本被 esversion: 6代替  
  使用ECMAScript 6具体语法，有些浏览器不支持
* [evil](http://jshint.com/docs/options/" \l "evil" \t "http://www.jianshu.com/p/_blank)  
  禁止使用eval
* [expr](http://jshint.com/docs/options/" \l "expr" \t "http://www.jianshu.com/p/_blank)  
  禁止使用表达式，一般的使用函数调用。
* [globalstrict](http://jshint.com/docs/options/" \l "globalstrict" \t "http://www.jianshu.com/p/_blank)  
  下个版本中会使用 strict: "global"代替。  
  全局严格模式会和第三方小插件冲突，所以不推荐使用。
* [lastsemic](http://jshint.com/docs/options/" \l "lastsemic" \t "http://www.jianshu.com/p/_blank)  
  检查一行代码最后声明后面的分号是否遗漏  
  var name = (function() { return 'Anton' }());
* [laxbreak](http://jshint.com/docs/options/" \l "laxbreak" \t "http://www.jianshu.com/p/_blank)  
  检查不安全的折行(下个版本将被删除)
* [laxcomma](http://jshint.com/docs/options/" \l "laxcomma" \t "http://www.jianshu.com/p/_blank)  
  检查逗号在代码行最前面的编程风格

var obj = {

name: 'Anton'

, handle: 'valueof'

, role: 'SW Engineer'

};

* [loopfunc](http://jshint.com/docs/options/" \l "loopfunc" \t "http://www.jianshu.com/p/_blank)  
  禁止内部循环，定义函数的内部循环可能导致这样的错误:

var nums = [];

for (var i = 0; i < 10; i++) {

nums[i] = function (j) {

return i + j;

};

}

nums[0](2); // Prints 12 instead of 2

* 解决上面的代码,你需要复制的变量 i:

var nums = [];

for (var i = 0; i < 10; i++) {

(function (i) {

nums[i] = function (j) {

return i + j;

};

}(i));

}

* [moz](http://jshint.com/docs/options/" \l "moz" \t "http://www.jianshu.com/p/_blank)  
  JSHint Mozilla扩展。除非你开发专门为Firefox web浏览器不需要这个选项。
* [multistr](http://jshint.com/docs/options/" \l "multistr" \t "http://www.jianshu.com/p/_blank)  
  将在下个版本中被删除  
  这个选项会抑制警告多行字符串。允许多行字符串在JavaScript是危险的,如果你小心在一个转义字符()和一个新行之间输入一个空格，将会导致整个字符串错误。  
  注意,即使这个选项允许正确多行字符串,它仍然警告说对多行字符串没有转义字符之间或与任何转义字符和空格。

// jshint multistr:true

var text = "Hello\World"; // All good.text = "HelloWorld"; // Warning, no escape character.text = "Hello\World"; // Warning, there is a space after \

* [noyield](http://jshint.com/docs/options/" \l "noyield" \t "http://www.jianshu.com/p/_blank)  
  检查函数生成器没有yield声明
* [plusplus](http://jshint.com/docs/options/" \l "plusplus" \t "http://www.jianshu.com/p/_blank)  
  禁止一元递增和递减运算符的使用
* [proto](http://jshint.com/docs/options/" \l "proto" \t "http://www.jianshu.com/p/_blank)  
  禁止关于\_\_proto\_\_属性的警告
* [scripturl](http://jshint.com/docs/options/" \l "scripturl" \t "http://www.jianshu.com/p/_blank)  
  禁止使用脚本URL定向，比如javascript:...
* [sub](http://jshint.com/docs/options/" \l "sub" \t "http://www.jianshu.com/p/_blank)  
  下个版本将被删除  
  检查[]使用，可以使用.代替[]  
  person['name'] vs.person.name.
* [supernew](http://jshint.com/docs/options/" \l "supernew" \t "http://www.jianshu.com/p/_blank)  
  检查怪异结构 new function () { ... }和 new Object;  
  这样的结构是有时用于单列在JavaScript中:

var singleton = new function() {

var privateVar;

this.publicMethod = function () {}

this.publicMethod2 = function () {}

};

* [validthis](http://jshint.com/docs/options/" \l "validthis" \t "http://www.jianshu.com/p/_blank)  
  ****注意:****可以使用这个选项只有在函数的范围  
  在非构造器函数中使用 this
* [withstmt](http://jshint.com/docs/options/" \l "withstmt" \t "http://www.jianshu.com/p/_blank)  
  检查with使用声明。  
  with声明语句可以引起开发者和意外全局变量定义之间的混乱。

### **环境变量**

这些选项让JSHint知道一些预先定义的全局变量。

* [browser](http://jshint.com/docs/options/" \l "browser" \t "http://www.jianshu.com/p/_blank)  
  暴露浏览器属性的全局变量，列如 window,document;  
  ****注意:****这个选项不暴露变量 alert或 console。
* [browserify](http://jshint.com/docs/options/" \l "browserify" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义全局变量使用时可用Browserify工具建立一个项目
* [couch](http://jshint.com/docs/options/" \l "couch" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义全局暴露CouchDB。CouchDB是一个面向文档的数据库,可以查询和索引MapReduce的方式使用JavaScript
* [devel](http://jshint.com/docs/options/" \l "devel" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义了全局变量,通常用于日志调试: console, alert等等
* [dojo](http://jshint.com/docs/options/" \l "dojo" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义全局暴露的Dojo Toolkit.
* [jasmine](http://jshint.com/docs/options/" \l "jasmine" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义全局暴露jasmine的单元测试框架.
* [jquery](http://jshint.com/docs/options/" \l "jquery" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义全局暴露的jQuery库。
* [mocha](http://jshint.com/docs/options/" \l "mocha" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义全局暴露的“BDD”和“TDD”的ui mocha单元测试框架.
* [module](http://jshint.com/docs/options/" \l "module" \t "http://www.jianshu.com/p/_blank)  
  这个选项告诉JSHint,输入代码描述了一个ECMAScript 6模块。所有模块的代码解释为严格模式代码。
* [mootools](http://jshint.com/docs/options/" \l "mootools" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义全局暴露的MooToolsJavaScript框架。
* [node](http://jshint.com/docs/options/" \l "node" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义全局变量可以当你的代码运行在node的运行时环境
* [nonstandard](http://jshint.com/docs/options/" \l "nonstandard" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义非标准但广泛采用全局变量等 escape和 unescape.
* [phantom](http://jshint.com/docs/options/" \l "phantom" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义全局可用你的核心运行时内部PhantomJS运行时环境
* [prototypejs](http://jshint.com/docs/options/" \l "prototypejs" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义全局暴露的prototypejs框架。
* [qunit](http://jshint.com/docs/options/" \l "qunit" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义全局暴露QUnit单元测试框架
* [rhino](http://jshint.com/docs/options/" \l "rhino" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义全局变量可以当你的代码运行在rhino的运行时环境。rhino是一个开源的实现完全用Java编写的JavaScript。
* [shelljs](http://jshint.com/docs/options/" \l "shelljs" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义全局暴露ShellJS库
* [typed](http://jshint.com/docs/options/" \l "typed" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义全局变量数组类型构造函数。
* [worker](http://jshint.com/docs/options/" \l "worker" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义全局变量可以当你的代码运行在web worker.web worker 是运行在后台的 JavaScript，独立于其他脚本，不会影响页面的性能.  
  ****所有主流浏览器均支持 web worker，除了 Internet Explorer。****
* [wsh](http://jshint.com/docs/options/" \l "wsh" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义全局变量可以当你的代码运行在Windows Script Host的运行时环境
* [yui](http://jshint.com/docs/options/" \l "yui" \t "http://www.jianshu.com/p/_blank)  
  这个选项定义全局暴露的yui框架。