ECMAScript 5增加了严格模式，严格模式可以:

消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为;

消除代码运行的一些不安全之处，保证代码运行的安全；

提高编译器效率，增加运行速度；

**1.调用**

进入严格模式的标识为 "use strict"； 严格模式有两种调用方法

①将"use strict"放在脚本文件的第一行，指定整个脚本都以严格模式运行

②将"use strict"放在函数体的第一行，指定函数以严格模式运行

**2.不同**

①正常模式中如果一个变量未经声明就赋值，会被认为是全局变量，而严格模式中，全局变量必须显式声明。

②严格模式在某些情况下只允许静态绑定

(1)禁止使用with语句;因为with语句无法在编译时就确定属性属于哪个对象

(2)正常模式只有全局作用域和函数作用域，而严格模式下增加了eval()作用域

严格模式下，eval语句本身就是一个作用域，它所生成的变量只能用于eval内部。

**3.安全措施**

①禁止this关键字指向全局对象

function f(){

　　　　return !this;

　　}

　　// 返回false，因为"this"指向全局对象，"!this"就是false

　　function f(){

　　　　"use strict";

　　　　return !this;

　　}

　　// 返回true，因为严格模式下，this的值为undefined，所以"!this"为true。

因此，在使用构造函数若没有加new,this不再指向全局对象，而会报错。

　function f(){

　　　　"use strict";

　　　　this.a = 1;

　　};

　　f();// 报错，this未定义

②禁止在函数内部遍历调用栈

　　function f1(){

　　　　"use strict";

　　　　f1.caller; // 报错

　　　　f1.arguments; // 报错

　　}

　　f1();

③禁止删除变量

严格模式下无法删除变量，除非configurable属性被设为true

"use strict";

　　var x;

　　delete x; // 语法错误

　　var o = Object.create(null, {'x': {

　　　　　　value: 1,

　　　　　　configurable: true

　　}});

　　delete o.x; // 删除成功

④显式报错

(1)正常模式对只读属性赋值不会报错，但严格模式会报错。

(2)严格模式下对一个使用getter方法读取的属性赋值会报错。

"use strict";

　　var o = {

　　　　get v() { return 1; }

　　};

　o.v = 2; // 报错

(3)严格模式下对禁止扩展的对象添加新属性会报错。

"use strict";

　　var o = {};

　　Object.preventExtensions(o);

　　o.v = 1; // 报错

(4)严格模式下删除一个不可删除的属性会报错。

4.重名错误

①严格模式对象不能有重名属性，正常模式取最后一个值，而严格模式为语法错误。

②严格模式函数不能有重名参数，正常模式可以通过arguments[i]读取

③严格模式禁止八进制表示法，整数第一位为0将会报错

④arguments对象限制

(1)不允许对arguments赋值

(2)arguments不再追踪参数的变化

　function f(a) {

　　　　a = 2;

　　　　return [a, arguments[0]];

　　}

　　f(1); // 正常模式为[2,2]

　　function f(a) {

　　　　"use strict";

　　　　a = 2;

　　　　return [a, arguments[0]];

　　}

　　f(1); // 严格模式为[2,1]

(3)禁止使用arguments.callee,意味着无法再匿名函数内部调用自身。

(4)函数必须声明在顶层。

5.保留字

为了向将来Javascript的新版本过渡，严格模式新增了一些保留字：implements, interface, let, package, private, protected, public, static, yield。

使用这些词作为变量名将会报错。

　　function package(protected) { // 语法错误

　　　　"use strict";

　　　　var implements; // 语法错误