没有块级作用域

//ES6有 let关键字定义变量

JavaScript 没有块级作用域经常会导致理解上的困惑。在其他类 C 的语言中，由花括号封闭的代码

块都有自己的作用域（如果用 ECMAScript 的话来讲，就是它们自己的执行环境），因而支持根据条件来定义变量。

```
if (true) {
    var color = "blue";
}

alert(color);    //"blue"
```

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

alert(i);      //10
```

1.声明变量

使用 var 声明的变量会自动被添加到最接近的环境中。在函数内部，最接近的环境就是函数的局部

环境；在 with 语句中，最接近的环境是函数环境。如果初始化变量时没有使用 var 声明，该变量会自

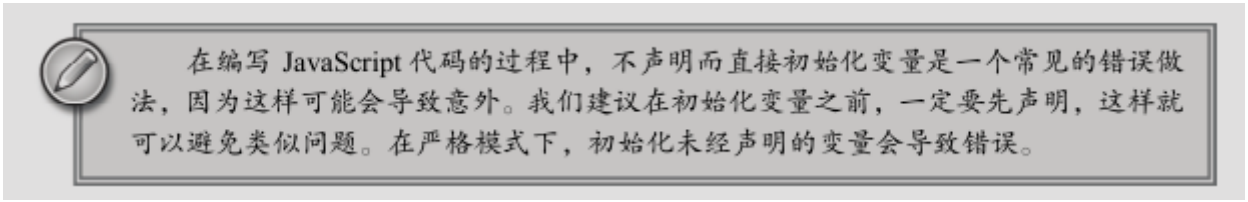
动被添加到全局环境。

```
function add(num1, num2) {
    var sum = num1 + num2;
    return sum;
}

var result = add(10, 20);  //30
alert(sum);                //由于 sum 不是有效的变量，因此会导致错误
```

```
function add(num1, num2) {
    sum = num1 + num2;
    return sum;
}

var result = add(10, 20);  //30
alert(sum);                //30
```



2. 查询标识符

按作用域链规则。

当在某个环境中为了读取或写入而引用一个标识符时，必须通过搜索来确定该标识符实际代表什

么。搜索过程从作用域链的前端开始，向上逐级查询与给定名字匹配的标识符。如果在局部环境中找到了该标识符，搜索过程停止，变量就绪。如果在局部环境中没有找到该变量名，则继续沿作用域链向上搜索。搜索过程将一直追溯到全局环境的变量对象。如果在全局环境中也没有找到这个标识符，则意味着该变量尚未声明。

```
var color = "blue";

function getColor(){
    return color;
}

alert(getColor());  //"blue"
```

在这个搜索过程中，如果存在一个局部的变量的定义，则搜索会自动停止，不再进入另一个变量对

象。换句话说，如果局部环境中存在着同名标识符，就不会使用位于父环境中的标识符

```
var color = "blue";

function getColor(){
    var color = "red";
    return color;
}

alert(getColor());  //"red"
```