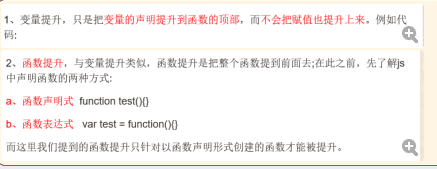
# 【[置顶] Javascript作用域和变量提升](http://blog.csdn.net/sunxing007/article/details/9034253)



下面的程序是什么结果？

**[javascript]** [view plain](http://blog.csdn.net/sunxing007/article/details/9034253" \o "view plain) [copy](http://blog.csdn.net/sunxing007/article/details/9034253" \o "copy) [print](http://blog.csdn.net/sunxing007/article/details/9034253" \o "print)[?](http://blog.csdn.net/sunxing007/article/details/9034253" \o "?)

1. var foo = 1;
2. function bar() {
3. if (!foo) {
4. var foo = 10;
5. }
6. alert(foo);
7. }
8. bar();

[IMG_259](http://blog.csdn.net/sunxing007/article/details/javascript:;)

var foo = 1;

function bar() {

if (!foo) {

var foo = 10;

}

alert(foo);

}

bar();

结果是10；

那么下面这个呢？

**[javascript]** [view plain](http://blog.csdn.net/sunxing007/article/details/9034253" \o "view plain) [copy](http://blog.csdn.net/sunxing007/article/details/9034253" \o "copy) [print](http://blog.csdn.net/sunxing007/article/details/9034253" \o "print)[?](http://blog.csdn.net/sunxing007/article/details/9034253" \o "?)

1. var a = 1;
2. function b() {
3. a = 10;
4. return;
5. function a() {}
6. }
7. b();
8. alert(a);

[IMG_260](http://blog.csdn.net/sunxing007/article/details/javascript:;)

var a = 1;

function b() {

a = 10;

return;

function a() {}

}

b();

alert(a);

结果是1.

吓你一跳吧？发生了什么事情？这可能是陌生的，危险的，迷惑的，同样事实上也是非常有用和印象深刻的**[JavaScript](http://lib.csdn.net/base/18" \o "JavaScript知识库" \t "http://blog.csdn.net/sunxing007/article/details/_blank)**语言特性。对于这种表现行为，我不知道有没有一个标准的称呼，但是我喜欢这个术语：“Hoisting (变量提升)”。这篇文章将对这种机制做一个抛砖引玉式的讲解，但是，首先让我们对javascript的作用域有一些必要的理解。

### ****Javascript的作用域****

对于Javascript初学者来说，一个最迷惑的地方就是作用域；事实上，不光是初学者。我就见过一些有经验的javascript程序员，但他们对scope理解不深。javascript作用域之所以迷惑，是因为它程序语法本身长的像C家族的语言，像下面的C程序：

**[cpp]** [view plain](http://blog.csdn.net/sunxing007/article/details/9034253" \o "view plain) [copy](http://blog.csdn.net/sunxing007/article/details/9034253" \o "copy) [print](http://blog.csdn.net/sunxing007/article/details/9034253" \o "print)[?](http://blog.csdn.net/sunxing007/article/details/9034253" \o "?)

1. #include <stdio.h>
2. int main() {
3. int x = 1;
4. printf("%d, ", x); // 1
5. if (1) {
6. int x = 2;
7. printf("%d, ", x); // 2
8. }
9. printf("%d\n", x); // 1
10. }

[IMG_261](http://blog.csdn.net/sunxing007/article/details/javascript:;)

#include <stdio.h>

int main() {

int x = 1;

printf("%d, ", x); // 1

if (1) {

int x = 2;

printf("%d, ", x); // 2

}

printf("%d\n", x); // 1

}

输出结果是1 2 1，这是因为C家族的语言有块作用域，当程序控制走进一个块，比如if块，只作用于该块的变量可以被声明，而不会影响块外面的作用域。但是在Javascript里面，这样不行。看看下面的代码：

**[javascript]** [view plain](http://blog.csdn.net/sunxing007/article/details/9034253" \o "view plain) [copy](http://blog.csdn.net/sunxing007/article/details/9034253" \o "copy) [print](http://blog.csdn.net/sunxing007/article/details/9034253" \o "print)[?](http://blog.csdn.net/sunxing007/article/details/9034253" \o "?)

1. var x = 1;
2. console.log(x); // 1
3. if (true) {
4. var x = 2;
5. console.log(x); // 2
6. }
7. console.log(x); // 2

[IMG_262](http://blog.csdn.net/sunxing007/article/details/javascript:;)

var x = 1;

console.log(x); // 1

if (true) {

var x = 2;

console.log(x); // 2

}

console.log(x); // 2

结果会是1 2 2。因为javascript是函数作用域。这是和c家族语言最大的不同。该程序里面的if并不会创建新的作用域。

对于很多C，c++，**[Java](http://lib.csdn.net/base/17" \o "Java EE知识库" \t "http://blog.csdn.net/sunxing007/article/details/_blank)**程序员来说，这不是他们期望和欢迎的。幸运的是，基于javascript函数的灵活性，这里有可变通的地方。如果你必须创建临时的作用域，可以像下面这样：

**[javascript]** [view plain](http://blog.csdn.net/sunxing007/article/details/9034253" \o "view plain) [copy](http://blog.csdn.net/sunxing007/article/details/9034253" \o "copy) [print](http://blog.csdn.net/sunxing007/article/details/9034253" \o "print)[?](http://blog.csdn.net/sunxing007/article/details/9034253" \o "?)

1. function foo() {
2. var x = 1;
3. if (x) {
4. (function () {
5. var x = 2;
6. // some other code
7. }());
8. }
9. // x is still 1.
10. }

[IMG_263](http://blog.csdn.net/sunxing007/article/details/javascript:;)

function foo() {

var x = 1;

if (x) {

(function () {

var x = 2;

// some other code

}());

}

// x is still 1.

}

这种方法很灵活，可以用在任何你想创建临时的作用域的地方。不光是块内。但是，我强烈推荐你花点时间理解javascript的作用域。它很有用，是我最喜欢的javascript特性之一。如果你理解了作用域，那么变量提升就对你显得更有意义。

### ****变量声明，命名，和提升****

在javascript，变量有4种基本方式进入作用域：

* 1 语言内置：所有的作用域里都有this和arguments；(译者注：经过测试arguments在全局作用域是不可见的)
* 2 形式参数：函数的形式参数会作为函数体作用域的一部分；
* 3 函数声明：像这种形式：function foo(){}；
* 4 变量声明：像这样：var foo；

函数声明和变量声明总是会被解释器悄悄地被“提升”到方法体的最顶部。这个意思是，像下面的代码：

**[javascript]** [view plain](http://blog.csdn.net/sunxing007/article/details/9034253" \o "view plain) [copy](http://blog.csdn.net/sunxing007/article/details/9034253" \o "copy) [print](http://blog.csdn.net/sunxing007/article/details/9034253" \o "print)[?](http://blog.csdn.net/sunxing007/article/details/9034253" \o "?)

1. function foo() {
2. bar();
3. var x = 1;
4. }

[IMG_264](http://blog.csdn.net/sunxing007/article/details/javascript:;)

function foo() {

bar();

var x = 1;

}

实际上会被解释成：

**[javascript]** [view plain](http://blog.csdn.net/sunxing007/article/details/9034253" \o "view plain) [copy](http://blog.csdn.net/sunxing007/article/details/9034253" \o "copy) [print](http://blog.csdn.net/sunxing007/article/details/9034253" \o "print)[?](http://blog.csdn.net/sunxing007/article/details/9034253" \o "?)

1. function foo() {
2. var x;
3. bar();
4. x = 1;
5. }

[IMG_265](http://blog.csdn.net/sunxing007/article/details/javascript:;)

function foo() {

var x;

bar();

x = 1;

}

无论定义该变量的块是否能被执行。下面的两个函数实际上是一回事：

**[javascript]** [view plain](http://blog.csdn.net/sunxing007/article/details/9034253" \o "view plain) [copy](http://blog.csdn.net/sunxing007/article/details/9034253" \o "copy) [print](http://blog.csdn.net/sunxing007/article/details/9034253" \o "print)[?](http://blog.csdn.net/sunxing007/article/details/9034253" \o "?)

1. function foo() {
2. if (false) {
3. var x = 1;
4. }
5. return;
6. var y = 1;
7. }
8. function foo() {
9. var x, y;
10. if (false) {
11. x = 1;
12. }
13. return;
14. y = 1;
15. }

[IMG_266](http://blog.csdn.net/sunxing007/article/details/javascript:;)

function foo() {

if (false) {

var x = 1;

}

return;

var y = 1;

}

function foo() {

var x, y;

if (false) {

x = 1;

}

return;

y = 1;

}

请注意，变量赋值并没有被提升，只是声明被提升了。但是，函数的声明有点不一样，函数体也会一同被提升。但是请注意，函数的声明有两种方式：

**[javascript]** [view plain](http://blog.csdn.net/sunxing007/article/details/9034253" \o "view plain) [copy](http://blog.csdn.net/sunxing007/article/details/9034253" \o "copy) [print](http://blog.csdn.net/sunxing007/article/details/9034253" \o "print)[?](http://blog.csdn.net/sunxing007/article/details/9034253" \o "?)

1. function test() {
2. foo(); // TypeError "foo is not a function"
3. bar(); // "this will run!"
4. var foo = function () { // 变量指向函数表达式
5. alert("this won't run!");
6. }
7. function bar() { // 函数声明 函数名为bar
8. alert("this will run!");
9. }
10. }
11. test();

[IMG_267](http://blog.csdn.net/sunxing007/article/details/javascript:;)

function test() {

foo(); // TypeError "foo is not a function"

bar(); // "this will run!"

var foo = function () { // 变量指向函数表达式

alert("this won't run!");

}

function bar() { // 函数声明 函数名为bar

alert("this will run!");

}

}

test();

这个例子里面，只有函数式的声明才会连同函数体一起被提升。foo的声明会被提升，但是它指向的函数体只会在执行的时候才被赋值。

上面的东西涵盖了提升的一些基本知识，它们看起来也没有那么迷惑。但是，在一些特殊场景，还是有一定的复杂度的。

### ****变量解析顺序****

最需要牢记在心的是变量解析顺序。记得我前面给出的命名进入作用域的4种方式吗？变量解析的顺序就是我列出来的顺序。(译者没理解这句，所以先做删除样式) **函数的声明比变量的声明具有高的优先级**。 (译者注: 关于函数的声明比变量的声明具有高的优先级,下面的程序能帮助你理解)

**[javascript]** [view plain](http://blog.csdn.net/sunxing007/article/details/9034253" \o "view plain) [copy](http://blog.csdn.net/sunxing007/article/details/9034253" \o "copy) [print](http://blog.csdn.net/sunxing007/article/details/9034253" \o "print)[?](http://blog.csdn.net/sunxing007/article/details/9034253" \o "?)

1. <script>
2. function a(){
3. }
4. var a;
5. alert(a);//打印出a的函数体
6. </script>
8. <script>
10. var a;
11. function a(){
12. }
13. alert(a);//打印出a的函数体
14. </script>
15. //但是要注意区分和下面两个写法的区别:
16. <script>
17. var a=1;
18. function a(){
19. }
20. alert(a);//打印出1
21. </script>
23. <script>
24. function a(){
25. }
27. var a=1;
29. alert(a);//打印出1
30. </script>

[IMG_268](http://blog.csdn.net/sunxing007/article/details/javascript:;)

<script>

function a(){

}

var a;

alert(a);//打印出a的函数体

</script>

<script>

var a;

function a(){

}

alert(a);//打印出a的函数体

</script>

//但是要注意区分和下面两个写法的区别:

<script>

var a=1;

function a(){

}

alert(a);//打印出1

</script>

<script>

function a(){

}

var a=1;

alert(a);//打印出1

</script>

这里有3个例外：  
1 内置的名称arguments表现得很奇怪，他看起来应该是声明在函数形式参数之后，但是却在函数声明之前。这是说，如果形参里面有arguments，它会比内置的那个有优先级。这是很不好的特性，所以要杜绝在形参里面使用arguments；  
2 在任何地方定义this变量都会出语法错误，这是个好特性；  
3 如果多个形式参数拥有相同的名称，最后的那个具有优先级，即便实际运行的时候它的值是undefined；

### 命名函数

你可以给一个函数一个名字。如果这样的话，它就不是一个函数声明，同时，函数体定义里面的指定的函数名( 如果有的话,如下面的spam, 译者注)将不会被提升,　而是被忽略。这里一些代码帮助你理解：

**[javascript]** [view plain](http://blog.csdn.net/sunxing007/article/details/9034253" \o "view plain) [copy](http://blog.csdn.net/sunxing007/article/details/9034253" \o "copy) [print](http://blog.csdn.net/sunxing007/article/details/9034253" \o "print)[?](http://blog.csdn.net/sunxing007/article/details/9034253" \o "?)

1. foo(); // TypeError "foo is not a function"
2. bar(); // valid
3. baz(); // TypeError "baz is not a function"
4. spam(); // ReferenceError "spam is not defined"
6. var foo = function () {}; // foo指向匿名函数
7. function bar() {}; // 函数声明
8. var baz = function spam() {}; // 命名函数，只有baz被提升，spam不会被提升。
10. foo(); // valid
11. bar(); // valid
12. baz(); // valid
13. spam(); // ReferenceError "spam is not defined"

[IMG_269](http://blog.csdn.net/sunxing007/article/details/javascript:;)

foo(); // TypeError "foo is not a function"

bar(); // valid

baz(); // TypeError "baz is not a function"

spam(); // ReferenceError "spam is not defined"

var foo = function () {}; // foo指向匿名函数

function bar() {}; // 函数声明

var baz = function spam() {}; // 命名函数，只有baz被提升，spam不会被提升。

foo(); // valid

bar(); // valid

baz(); // valid

spam(); // ReferenceError "spam is not defined"

### 怎么写代码

现在你理解了作用域和变量提升，那么这对于javascript编码意味着什么？最重要的一点是，总是用var定义你的变量。而且我强烈推荐，对于一个名称，在一个作用域里面永远只有一次var声明。如果你这么做，你就不会遇到作用域和变量提升问题。

### 语言规范怎么说

我发现ECMAScript参考文档总是很有用。下面是我找到的关于作用域和变量提升的部分：  
如果变量在函数体类声明，则它是函数作用域。否则，它是全局作用域（作为global的属性）。变量将会在执行进入作用域的时候被创建。块不会定义新的作用域，只有函数声明和程序（译者以为，就是全局性质的代码执行）才会创造新的作用域。变量在创建的时候会被初始化为undefined。如果变量声明语句里面带有赋值操作，则赋值操作只有被执行到的时候才会发生，而不是创建的时候。

我期待这篇文章会对那些对javascript比较迷惑的程序员带来一丝光明。我自己也尽最大的可能去避免带来更多的迷惑。如果我说错了什么，或者忽略了什么，请告知。

### 译者补充

[xu281828044](http://blog.csdn.net/xu281828044" \t "http://blog.csdn.net/sunxing007/article/details/_blank)提醒了我发现了IE下全局作用域下命名函数的提升问题:  
我翻译文章的时候是这么测试的:

**[html]** [view plain](http://blog.csdn.net/sunxing007/article/details/9034253" \o "view plain) [copy](http://blog.csdn.net/sunxing007/article/details/9034253" \o "copy) [print](http://blog.csdn.net/sunxing007/article/details/9034253" \o "print)[?](http://blog.csdn.net/sunxing007/article/details/9034253" \o "?)

1. <script>
2. functiont(){
3. spam();
4. var baz = function spam() {alert('this is spam')};
5. }
6. t();
7. </script>

[IMG_270](http://blog.csdn.net/sunxing007/article/details/javascript:;)

<script>

functiont(){

spam();

var baz = function spam() {alert('this is spam')};

}

t();

</script>

这种写法, 即非全局作用域下的命名函数的提升,在ie和ff下表现是一致的. 我改成:

**[html]** [view plain](http://blog.csdn.net/sunxing007/article/details/9034253" \o "view plain) [copy](http://blog.csdn.net/sunxing007/article/details/9034253" \o "copy) [print](http://blog.csdn.net/sunxing007/article/details/9034253" \o "print)[?](http://blog.csdn.net/sunxing007/article/details/9034253" \o "?)

1. <script>
2. spam();
3. var baz = function spam() {alert('this is spam')};
4. </script>

[IMG_271](http://blog.csdn.net/sunxing007/article/details/javascript:;)

<script>

spam();

var baz = function spam() {alert('this is spam')};

</script>

则ie下是可以执行spam的,ff下不可以. 说明不同浏览器在处理这个细节上是有差别的.

这个问题还引导我思考了另2个问题，1:**对于全局作用于范围的变量，var与不var是有区别的.** 没有var的写法，其变量不会被提升。比如下面两个程序，第二个会报错：

**[html]** [view plain](http://blog.csdn.net/sunxing007/article/details/9034253" \o "view plain) [copy](http://blog.csdn.net/sunxing007/article/details/9034253" \o "copy) [print](http://blog.csdn.net/sunxing007/article/details/9034253" \o "print)[?](http://blog.csdn.net/sunxing007/article/details/9034253" \o "?)

1. <script>
2. alert(a);
3. var a=1;
4. </script>

[IMG_272](http://blog.csdn.net/sunxing007/article/details/javascript:;)

<script>

alert(a);

var a=1;

</script>

**[html]** [view plain](http://blog.csdn.net/sunxing007/article/details/9034253" \o "view plain) [copy](http://blog.csdn.net/sunxing007/article/details/9034253" \o "copy) [print](http://blog.csdn.net/sunxing007/article/details/9034253" \o "print)[?](http://blog.csdn.net/sunxing007/article/details/9034253" \o "?)

1. <script>
2. alert(a);
3. a=1;
4. </script>

[IMG_273](http://blog.csdn.net/sunxing007/article/details/javascript:;)

<script>

alert(a);

a=1;

</script>

2: **eval中创建的局部变量是不会被提升的(它也没办法做到).**

**[html]** [view plain](http://blog.csdn.net/sunxing007/article/details/9034253" \o "view plain) [copy](http://blog.csdn.net/sunxing007/article/details/9034253" \o "copy) [print](http://blog.csdn.net/sunxing007/article/details/9034253" \o "print)[?](http://blog.csdn.net/sunxing007/article/details/9034253" \o "?)

1. <script>
2. var a = 1;
3. function t(){
4. alert(a);
5. eval('var a = 2');
6. alert(a);
7. }
8. t();
9. alert(a);
10. </script>

[IMG_274](http://blog.csdn.net/sunxing007/article/details/javascript:;)

<script>

var a = 1;

function t(){

alert(a);

eval('var a = 2');

alert(a);

}

t();

alert(a);

</script>

原文地址：http://www.adequatelygood.com/JavaScript-Scoping-and-Hoisting.html

转载请注明来自: http://blog.csdn.net/sunxing007