  一直对Js的作用域有点迷糊，今天偶然读到Javascript权威指南，立马被吸引住了，写的真不错。我看的是第六版本，相当的厚，大概1000多页，Js博大精深，要熟悉精通需要大毅力大功夫。

一：函数作用域

   先看一小段代码：

**[javascript]** [view plaincopy](http://blog.csdn.net/yueguanghaidao/article/details/9568071)

1. **var** scope="global";
2. **function** t(){
3. console.log(scope);
4. **var** scope="local"
5. console.log(scope);
6. }
7. t();

(PS: console.log()是firebug提供的调试工具，很好用，有兴趣的童鞋可以用下，比浏览器+alert好用多了）

第一句输出的是： "undefined"，而不是 "global"

第二讲输出的是："local"

  你可能会认为第一句会输出："global",因为代码还没执行var scope="local",所以肯定会输出“global"。

  我说这想法完全没错，只不过用错了对象。我们首先要区分Javascript的函数作用域与我们熟知的C/C++等的块级作用域。

  在C/C++中，花括号内中的每一段代码都具有各自的作用域，而且变量在声明它们的代码段之外是不可见的。而Javascript压根没有块级作用域，而是函数作用域.

所谓函数作用域就是说：-》变量在声明它们的函数体以及这个函数体嵌套的任意函数体内都是有定义的。

所以根据函数作用域的意思，可以将上述代码重写如下：

**[javascript]** [view plaincopy](http://blog.csdn.net/yueguanghaidao/article/details/9568071)

1. **var** scope="global";
2. **function** t(){
3. **var** scope;
4. console.log(scope);
5. scope="local"
6. console.log(scope);
7. }
8. t();

    我们可以看到，由于函数作用域的特性，局部变量在整个函数体始终是由定义的，我们可以将变量声明”提前“到函数体顶部，同时变量初始化还在原来位置。

为什么说Js没有块级作用域呢，有以下代码为证：

**[javascript]** [view plaincopy](http://blog.csdn.net/yueguanghaidao/article/details/9568071)

1. **var** name="global";
2. **if**(**true**){
3. **var** name="local";
4. console.log(name)
5. }
6. console.log(name);

都输出是“local",如果有块级作用域，明显if语句将创建局部变量name,并不会修改全局name,可是没有这样，所以Js没有块级作用域。

现在很好理解为什么会得出那样的结果了。scope声明覆盖了全局的scope,但是还没有赋值，所以输出：”undefined“。

所以下面的代码也就很好理解了。

**[javascript]** [view plaincopy](http://blog.csdn.net/yueguanghaidao/article/details/9568071)

1. **function** t(flag){
2. **if**(flag){
3. **var** s="ifscope";
4. **for**(**var** i=0;i<2;i++)
5. ;
6. }
7. console.log(i);
8. console.log(s);
9. }
10. t(**true**);

输出：2  ”ifscope"

二：变量作用域

还是首先看一段代码：

**[javascript]** [view plaincopy](http://blog.csdn.net/yueguanghaidao/article/details/9568071)

1. **function** t(flag){
2. **if**(flag){
3. s="ifscope";
4. **for**(**var** i=0;i<2;i++)
5. ;
6. }
7. console.log(i);
8. }
9. t(**true**);
10. console.log(s);

就是上面的翻版，知识将声明s中的var去掉。

程序会报错还是输出“ifscope"呢？

让我揭开谜底吧，会输出：”ifscope"

这主要是Js中没有用var声明的变量都是全局变量，而且是顶层对象的属性。

所以你用console.log(window.s)也是会输出“ifconfig"

当使用var声明一个变量时，创建的这个属性是不可配置的，也就是说无法通过delete运算符删除

var name=1    ->不可删除

sex=”girl“         ->可删除

this.age=22    ->可删除

三：作用域链

先来看一段代码：

**[javascript]** [view plaincopy](http://blog.csdn.net/yueguanghaidao/article/details/9568071)

1. name="lwy";
2. **function** t(){
3. **var** name="tlwy";
4. **function** s(){
5. **var** name="slwy";
6. console.log(name);
7. }
8. **function** ss(){
9. console.log(name);
10. }
11. s();
12. ss();
13. }
14. t();

当执行s时，将创建函数s的执行环境(调用对象),并将该对象置于链表开头，然后将函数t的调用对象链接在之后，最后是全局对象。然后从链表开头寻找变量name,很明显

name是"slwy"。

但执行ss()时，作用域链是： ss()->t()->window,所以name是”tlwy"

下面看一个很容易犯错的例子：

**[html]** [view plaincopy](http://blog.csdn.net/yueguanghaidao/article/details/9568071)

1. **<html>**
2. **<head>**
3. **<script** type="text/javascript"**>**
4. function buttonInit(){
5. for(var i=1;i**<4**;i++){
6. var b=document.getElementById("button"+i);
7. b.addEventListener("click",function(){ alert("Button"+i);},false);
8. }
9. }
10. window.onload=buttonInit;
11. **</script>**
12. **</head>**
13. **<body>**
14. **<button** id="button1"**>**Button1**</button>**
15. **<button** id="button2"**>**Button2**</button>**
16. **<button** id="button3"**>**Button3**</button>**
17. **</body>**
18. **</html>**

当文档加载完毕，给几个按钮注册点击事件，当我们点击按钮时，会弹出什么提示框呢？

很容易犯错，对是的，三个按钮都是弹出："Button4",你答对了吗？

当注册事件结束后，i的值为4，当点击按钮时，事件函数即function(){ alert("Button"+i);}这个匿名函数中没有i,根据作用域链，所以到buttonInit函数中找，此时i的值为4，

所以弹出”button4“。

四：with语句

说到作用域链，不得不说with语句。with语句主要用来临时扩展作用域链，将语句中的对象添加到作用域的头部。

看下面代码

**[javascript]** [view plaincopy](http://blog.csdn.net/yueguanghaidao/article/details/9568071)

1. person={name:"yhb",age:22,height:175,wife:{name:"lwy",age:21}};
2. **with**(person.wife){
3. console.log(name);
4. }

with语句将person.wife添加到当前作用域链的头部，所以输出的就是：“lwy".

with语句结束后，作用域链恢复正常。