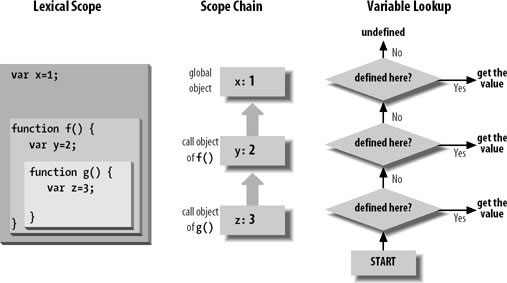
[深入理解JS中的变量作用域](http://blog.csdn.net/beijiguangyong/article/details/8301707)

在JS当中一个变量的作用域（scope）是程序中定义这个变量的区域。变量分为两类：全局（global）的和局部的。其中全局变量的作用域是全局性的，即在[JavaScript](http://lib.csdn.net/base/javascript)代码中，它处处都有定义。而在函数之内声明的变量，就只在函数体内部有定义。它们是局部变量，作用域是局部性的。函数的参数也是局部变量，它们只在函数体内部有定义。

我们可以借助JavaScript的作用域链（scope chain）更好地了解变量的作用域。每个JavaScript执行环境都有一个和它关联在一起的作用域链。这个作用域链是一个对象列表或对象链。当JavaScript代码需要查询变量x的值时（这个过程叫做变量解析（variable name resolution）），它就开始查看该链的第一个对象。如果那个对象有一个名为x的属性，那么就采用那个属性的值。如果第一个对象没有名为x的属性，JavaScript就会继续查询链中的第二个对象。如果第二个对象仍然没有名为x的属性，那么就继续查询下一个对象，以此类推。如果查询到最后（指顶层代码中）不存在这个属性，那么这个变量的值就是未定义的。



以上的过程并不是我们所熟悉的顺序结构，但大致与顺序结构类似只不过是将作用域当作一个整体来看待而已。整个过程如上图所示

JS作用域实例

代码一

var i=10;

function a() {

    alert(i);

};

a();

代码二

var i=10;

function a() {

    alert(i);

    var i = 2;

};

a();

根据“多年”的编程经验你可能觉得这两个代码输出是一样的，但是事实却是第一个代码正常输出了变量的值----10，而第二个代码输出的却是undefined。也许很多人理解不了，但是根据前面的作用域链的图我们就很好理解了。

作用域链图中很明确的表示出：在变量解析过程中首先查找局部的作用域，然后查找上层作用域。在代码一的函数当中没有定义变量i，于是查找上层作用域（全局作用域），进而进行输出其值。但是在代码二的函数内定义了变量i（无论是在alter之后还是之前定义变量，都认为在此作用域拥有变量i），于是不再向上层的作用域进行查找，直接输出i。但是不幸的是此时的局部变量i并没有赋值，所以输出的是undefined。

《JavaScript权威指南》中提出的“没有块级作用域”实际上就是上述的意思。很多时候认为懂了、理解了，其实没有懂，细细的研究一番之后看回过头来再书中那加粗的文字，顿时恍然大悟，其实人家书里说的挺清楚的嘛！