1. 变量作用域（38页）
2. 尽量少用全局变量：
3. 始终声明局部变量：

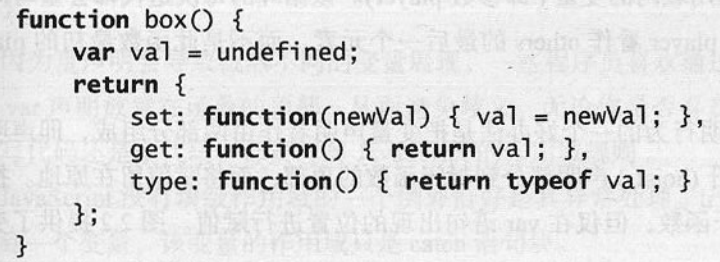
避免意外创建全局变量。

1. 避免使用with：
2. 熟练地使用闭包：

第一：闭包运行在外部函数定义变量。

第二：外部函数即使已经返回，闭包依然可以引用外部函数的变量。

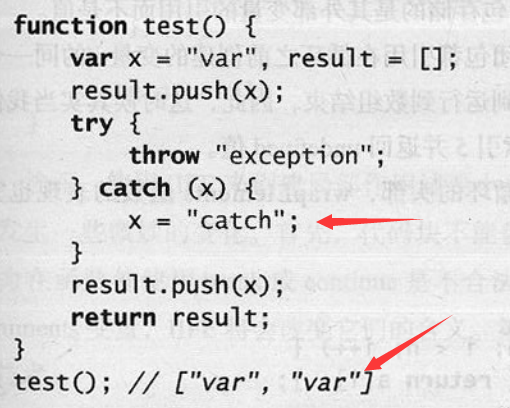
第三：闭包可以更新外部变量的值。



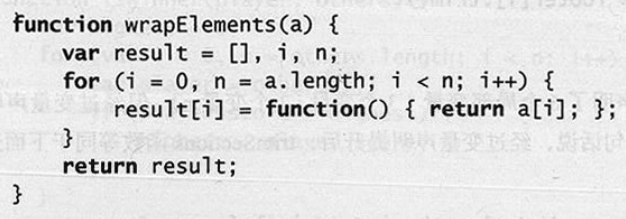
经典私有变量。

1. 理解变量声明提升：

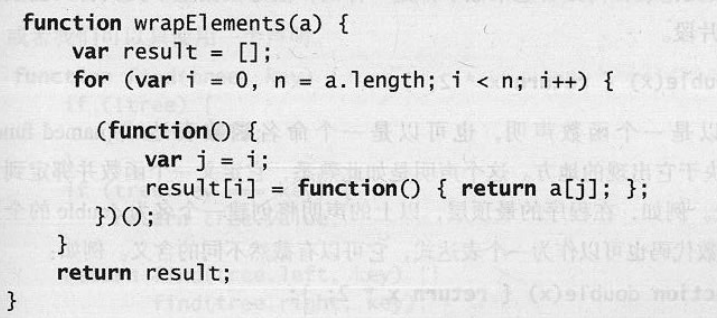
Es6之前都没有块级作用域，除了try，catch



1. 使用立即调用函数定义局部变量：



函数return的时候的i是外部闭包里面的i，因此引用的都是同一个值。



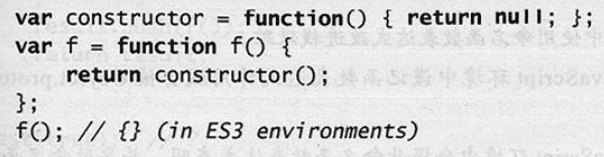
每次for循环的时候都单独创建了一个闭包，函数return的时候的j是外部闭包里面的j。

1. 当心命名函数表达式笨拙的作用域：



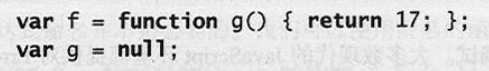
该语句将函数绑定在f变量上，而不是double。而double会作为一个函数内的局部变量。

但是使用double来声明函数似乎没有必要，f同样可以表示这个函数。



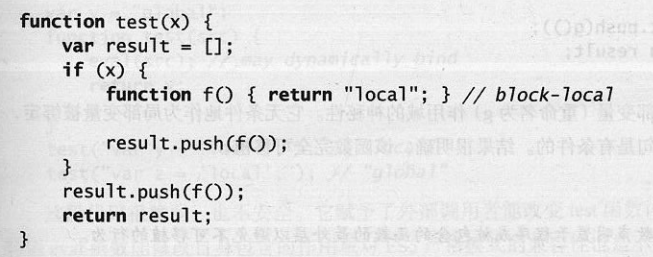
这里返回{}的原因是 命名函数表达式 的作用域是Object，就像with函数一样。

有的地方为命名函数分配了单独的空间，导致占用内存，下面f和g不是同一个对象。

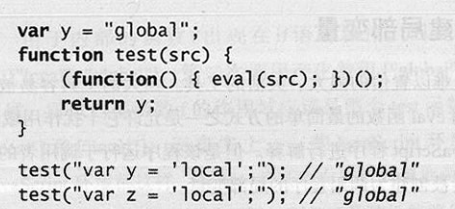


应该避免使用命名函数表达式

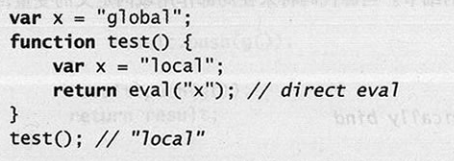
1. 当心局部块函数声明笨拙的作用域



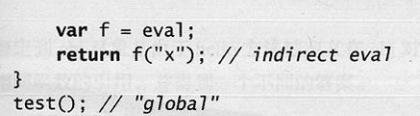
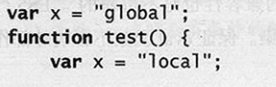
1. 避免使用eval创建局部变量：



1. 间接调用eval：



直接调用，可以访问到局部变量



间接调用，只能访问到全局变量。

简写的间接调用。

通常情况下，直接调用会暴露函数的内部细节，消耗的性能更加多。