先说一下变量，变量包括基本类型和引用类型，

我们基本的五种数据类型，number string Boolean null undefined ，这些都是基本类型，他们都是按值传递的，比如说，

Var text = 5;

Var test1 = tese;

这个时候test1就会从text那里复制一个5给自己，自己怎么玩儿都不会影响到text，这个就是按值传递。

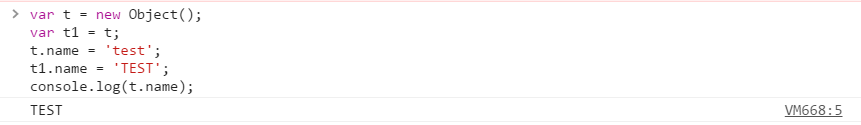
引用类型指那些可能由多个值构成的对象，比如说object

引用类型是按照引用访问的，比如说

Var t = new Object();

Var t1 = t;

这时候，tl得到的只是一个指向t在存储在堆内存中对象的一个指针，这时候我们操作t1的话，t也是会跟着改变的，足以证明，引用类型传递的是引用。



-------------------------------------------------------------

再说一下传递参数，如果上面理解了，这里只需要记住一点就可以了，就是参数的传递永远都是按值传递的，不管是基本类型还是引用类型。

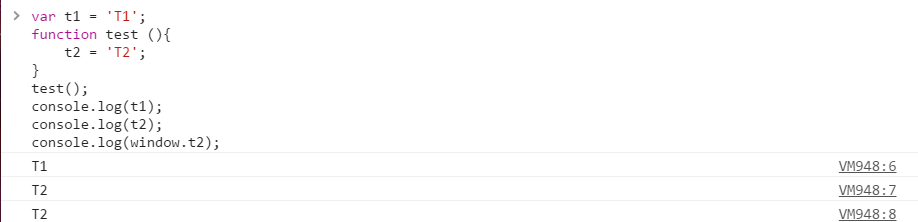
这里我觉得这样理解比较好，无论是值还是地址（指针），都是值。

全局作用域和局部作用域，全局作用域，在全局任何一个地方都能访问到，而局部作用域一般只在特定的位置才能访问到，比如说函数的内部

全局作用域一般包涵下面三种情况，

1. 在最外层定义的变量和函数， //这个一般大家都理解
2. 没有声明，直接赋值的变量 //这个看下面的例子

仔细理解了一下，就是说js里面没有声明的变量都是全局变量，而且是顶层window对象。



1. Window对象的内置属性

思路有点乱，

函数作用域，变量在声明他的函数体以及这个函数体内嵌套的函数体内都是有定义的，

由此我竟然想到了闭包。。。

下面说一说执行环境和作用域链

执行环境（执行上下文）

1，在执行代码之前，会把要用到的所有变量都事先拿出来，有的会直接赋值，有的会用undefind占个位置。

2，函数在定义的时候就已经确定了内部变量的作用域。3，函数没执行一次，都会产生一个新的执行上下文环境。

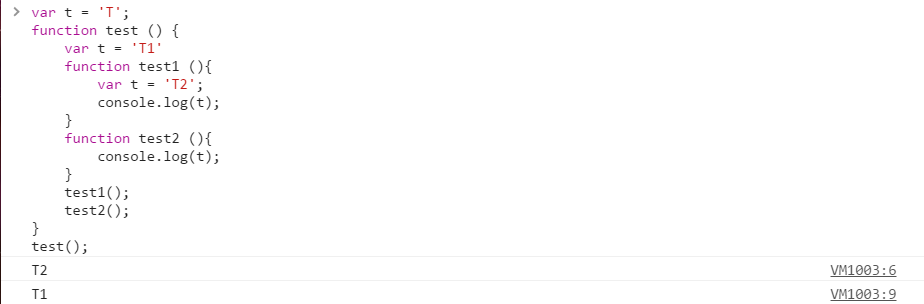
1. 执行上下文定义了变量或者函数有权访问的其他数据，比较抽象，我也没有进行深入的了解，但是有一些还是要知道的，执行环境实在一个栈里面，栈底是全局执行环境，栈顶是当前执行环境，当执行完当前的执行环境之后，出栈，然后将执行环境交给上一层。Window是最外层的执行环境。
2. 加强理解，首先会有一个全局上下文环境，当代码执行到函数的时候，再为该函数创建执行环境，这样就形成了一个执行上下文栈，正在执行的环境，就是活动对象，执行完，销毁，出栈，然后将执行环境交给栈里面的下一个，王福朋大神描述的过程非常好，在这里附上博客地址http://www.cnblogs.com/wangfupeng1988/p/3989357.html，

作用域链

当一个函数在执行环境中执行的时候，会创建一个作用域链，用来保证，将有权访问的对象进行有序的访问，

其实就是一个寻找我所需要的变量的过程，当前函数作用域内找不到的话就到创建这个函数的作用域内去找

当我们执行一个函数时，会先创建一个该函数的执行环境，下面看例子



但我们执行函数test1的时候，创建函数test1的执行环境，并将该函数置于链表头部，然后是函数test调用的对象，再然后是全局对象window

作用域链是test1（）🡪test()🡪window

当我们调用test1（）；的时候，t这个对象会现在函数内部找，找到了，就返回，不会再往外面找了。

看到这里，其实很多以前不明白的地方就豁然开朗了，比如说函数内部的变量和全局变量同名的时候，调用该函数时，函数内部的局部变量会覆盖掉全局变量。

我们再看一个例子（这个是直接复制过来的）



这里很容易犯错，当我们依次点击按钮1，2，3时会输出什么？

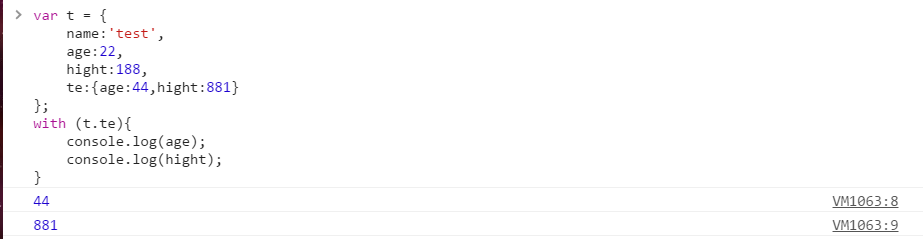
答案是 button4 button4 button4

为啥呢？ 页面加载完之后，我们就会一次给三个按钮添加监听器，

当我们点击按钮的时候，会触发监听的点击事件，这时候调用预先设定好的回调函数

这个函数会输出 button+i 问题就出现在i上面，当我们执行完添加监听事件之后，这时候i就已经是4了，当执行回调函数的时候，function(){}本身没有i这个变量，根据作用域链，会往上层函数找，此时i为4，找到了，直接返回，所以点哪个按钮都会输出button4

最后一点，with语句，他主要是用该改变作用域链的顺序，看例子



其实就是说，with把t.te放到了作用域链的头部

暂时先写这么多吧，后面深入了解的时候再面满更新。