**神奇的“连续”**

<http://www.silverna.org/blog/?p=73>  
  
先看一段代码：

[[复制]](http://bbs.51js.com/viewthread.php?tid=85325&highlight=###)

**Code:**

Function.prototype.$continuous = function(fn){  
    var me = this;  
    return function(){  
        var currentArgs = Array.prototype.slice.call(arguments, 0, me.length);  
        var moreArgs = Array.prototype.slice.call(arguments, me.length);  
  
        ret = me.apply(this, currentArgs);  
  
        if(moreArgs.length > 0){  
            ret = fn.call(this, arguments.callee, ret, moreArgs);  
        }  
  
        return ret;  
    }  
}

这个函数并不复杂，它的作用是包装一个函数，判断它实际调用的参数个数和形参个数，当实际调用的参数个数大于形参个数时，再用一个闭包进行后续操作，这个闭包有3个参数，分别是包装的函数自身、前次调用的返回值，以及多余的参数。  
  
这个简单的函数其实比想象得有用：

[[复制]](http://bbs.51js.com/viewthread.php?tid=85325&highlight=###)

**Code:**

var add = function(x,y){  
    return x+y;  
}

add显然只是一个简单的两个数相加的程序，如果想让它支持多个数相加呢？

[[复制]](http://bbs.51js.com/viewthread.php?tid=85325&highlight=###)

**Code:**

function reducer(target, returnValue, moreArgs){  
    return target.apply(this, [returnValue].concat(moreArgs));  
}  
add = add.$continuous(reducer);  
var a = add(1,2,3,4,5);  //a = 1+2+3+4+5=15

同样的：

[[复制]](http://bbs.51js.com/viewthread.php?tid=85325&highlight=###)

**Code:**

var max = function(x,y){  
    return x>y?x:y;  
}  
max = max.$continuous(reducer);  
var a = max(1,2,3,2,1); //a=3

还有别的作用：

[[复制]](http://bbs.51js.com/viewthread.php?tid=85325&highlight=###)

**Code:**

function processor(target, returnValue, moreArgs){  
    return [returnValue].concat(target.apply(this, moreArgs));  
}     
var $ = function(id){  
    return document.getElementById(id);  
}.$continuous(processor);  
var els = $("a","b","c");  //得到3个elements（返回数组）

最后总结一下：  
$continuous本身很简单，但是它可以被reducer或processor作用，也就是说，返回结果可以被push，也可以被reduce，上面的reducer实际上是先push再reduce，后面的processor我改了一个次序，先reduce再push，就能获得截然不同的用途。  
  
不知道通过上面的叙述，大家对脚本库核心的基本设计和函数式(functional)编程的思想方法有没有新的认识。不管怎样，JavaScript灵活多变的特性，总能让前端开发充满乐趣。 :)