说出它们的值  
  
1、typeof(NaN) 、typeof(Infinity)、typeof(null)、typeof(undefined)  
2、NaN == NaN   
3、NaN != NaN  
4、NaN >= NaN  
5、null == undefined  
6、null >= undefined  
7、null <= undefined  
8、parseInt("123abc")  
9、"123abc" - 0   
10、Infinity > 10  
11、Infinity > "abc"  
12、Infinity == NaN  
13、true == 1  
14、new String("abc") == "abc"  
15、new String("abc") === "abc"

**Number 对象**

除了 **Math** 对象中可用的特殊数值属性（例如 **PI**）外，在 Microsoft Jscript 中， **Number** 对象有几个其他的数值属性。

|  |  |
| --- | --- |
| **属性** | **描述** |
| MAX\_VALUE | 可能的最大数大约为 1.79E+308；可以是正数或负数。（具体数值随系统不同而有微小差别。） |
| MIN\_VALUE | 可能的最小数大约为 2.22E-308；可以是正数或负数。（具体数值随系统不同而有微小差别。） |
| NaN | 特殊非数量值，“不是数值”。 |
| POSITIVE\_INFINITY | 比最大正数（Number.MAX\_VALUE）还要大的任意正值自动被转换为此值，代表正无穷大。 |
| NEGATIVE\_INFINITY | 比最小的负数（负的 Number.MAX\_VALUE）还小的任意值被自动转换为此值，代表负无穷。 |

也就是说 Number. POSITIVE\_INFINITY 返回Infinity 比 Number.MAX\_VALUE还要大

Number.NEGATIVE\_INFINITY = -Number.POSITIVE\_INFINITY = -Infinity

null 是空对象，undefined是另一种类型，在转换成boolean时值相等

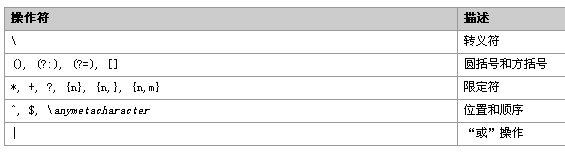
new String("abc") === "abc" 值虽然相等，但它们所拥有的、可用的方法并不相同。

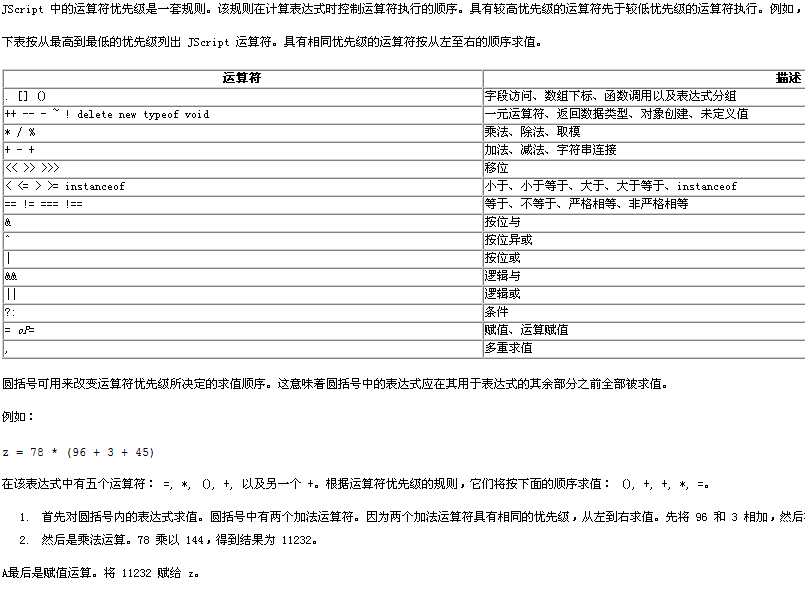
说出它们的输出结果  
  
1、  
var a = "123abc";  
alert(typeof(a++));//NaN是number的一种类型  
alert(a);  
  
2、  
var a = "123abc";  
a.valueOf = function(){return parseInt(a);}//依旧是NaN  
alert(++a);  
alert(a-0);  
  
3、  
var a = new Object();  
a.toString = function(){return "123abc";}  
a.valueOf = function(){return parseInt(a);}  
alert(++a);  
alert(a-0);

//选调用valueOf 然后再调用toString 仅对象可会进行此类型的转换（javascript编程精髓有讲解）  
  
4、  
String.prototype.valueOf = function()  
{  
    return parseFloat(this);  
}  
alert("123abc" > 122);  
alert(new String("123abc") > 122);

//对象会调用valueOf 字符串并不进行调用  
  
5、  
var s = new String("abc");  
alert(typeof(s) == typeof("abc"));//objec 与 string  
alert(s === "abc");//valueOf的值倒是一致的  
alert(s.toString() == s);//显示调用也是可以的  
  
6、  
var a = new Object();  
a.toString = function(){return "a"};  
var b = new Object();  
b.toString = function(){return "b"};  
alert(a>b);//false  
a.valueOf = function(){return 1};  
b.valueOf = function(){return 0};  
alert(a>b);//true 得好好再研究了  
  
7、  
function step(a)  
{  
    return function(x)  
    {  
        return x + a++;  
    }  
}  
var a = step(10);  
var b = step(20);  
alert(a(10));//20  
alert(b(10));//30

去算符的优先级问题：





看来正确的规则应当是  
如果函数返回的是一个值类型的对象，那么new表达式的值仍然是this所引用的对象，函数的返回值将被舍弃。否则，如果函数返回的是一个引用类型的对象，那么new表达式的值将被返回的对象所取代，this所引用的对象被抛弃  
  
这个可能和JavaScript实现时使用函数栈空间的实现算法有关系！