运算中的隐式类型转换

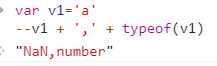
1.递增递减操作符

递增“++” 、 递减“--” 运算符不仅适用于**整数**，还可以用于**字符串**、**布尔值**、**浮点数值**和**对象**。

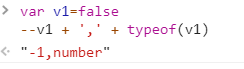
（1）是包含有效数字字符的字符串时，先将其**转换为数字值**，再进行运算



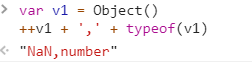
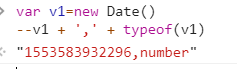
（2）是不包含有效数字字符的字符串时，将其值**转换为NaN**



（3）是布尔值时，false转换为数值**0**，true转换为数值**1**



（4）是对象时，先调用对象的valueOf()方法，以取得一个可供操作的值，如果结果是NaN，调用Number()或者toString()方法



2.一元加减操作符

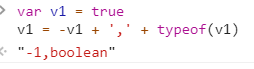
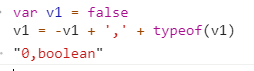
一元加号（+）放在**数值**前面，对数值**不会**产生任何影响。

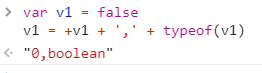


在对**非数值**应用一元加减操作符时，该操作符会像**Number()转型**函数一样对**这个值(最后结果不是中间值)**执行转换。

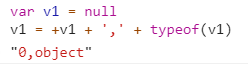
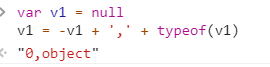
Number类型的转换规则如下：

（1）如是布尔值，true转换为1，false转换为0

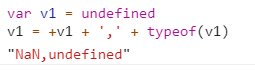




（2）如是null，返回0



（3）如是undefined，返回NaN

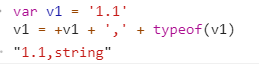


（4）如是字符串，规则

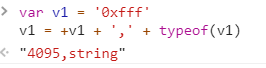
如字符串中只包含数字（包括前面带正负号的情况），将其转换为十进制数值



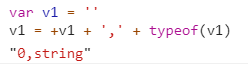
如字符串中包含有效的浮点格式，将其转换为对应的浮动数值



如字符串中包含有效的十六进制格式，例如“oxfff”，将其转换为相同大小的十进制整数值



如字符串为空，转换为0



如字符中包含除上述格式之外的字符，将其转换为NaN

3.加法

（1）如果两个操作数都是数值，执行常规的加法计算

（2）如有一个操作数是NaN，结果是NaN



（3）Infinity + Infinity=Infinity



（4）-Infinity + -Infinity = -Infinity



（5）Infinity+ -Infinity= NaN

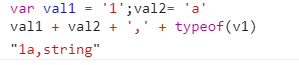


（6）0+0=0

（7）如果两个操作数都是字符串，将两个字符串拼接



（8）如果只有一个操作数是**字符串**，将另一个操作数转换为字符串，然后再将两个字符串拼接



（9）如果有一个操作数是**对象**、**数值**或**布尔值**，则调用它们的**toString()方法**取得相应的字符串值，在应用上述的字符串规则。对于undefined和null，则分别调用String()函数并取得字符串"**undefined**"和"**null**"

4.减法

（1）如果两个操作数都是数值，执行常规的减法计算

（2）如有一个操作数是NaN，结果是NaN



（3）Infinity-Infinity=NaN (同号)



（4）-Infinity - -Infinity = NaN (同号)



（5）Infinity- -Infinity= Infinity (异号)



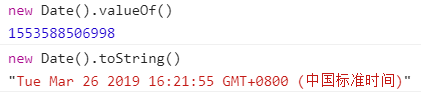
（6）-Infinity- Infinity=-Infinity (异号)

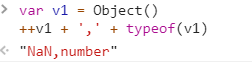
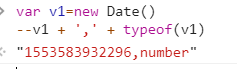


（7）0-0=0

（8）如果有一个操作数是**字符串**，**布尔值**、**null**或**undefined**，先在后台调用Number()函数将其转换为数值，再计算，如果转换结果是NaN，则结果是NaN

（9）如果有一个操作数是**对象**，则调用对象的valueOf()方法以取得表示该对象的数值，如果得到的值是NaN，则结果就是NaN。如果对象没有**valueOf()**方法，则调用**Number()**或者**toString()**方法

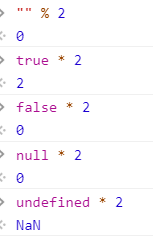




5.乘性操作符

**乘法、除法、求模**

如果参与乘性操作数不是数值，后台会先使用Number()转型函数将其转换为数值。也就是说**空字符串**转为**0**，**true**转为**1**，**flase**转为**0**， **null**转为**0**， undefined转为**NaN**



6.关系操作符

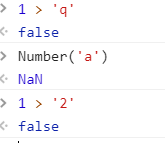
关系操作符：<  >   <=   >=

（1）如两个操作数都是**数值**，则执行数值比较

（2）如两个操作数都是**字符串**，则比较两个字符串对应的字符编码值



（3）如一个操作数是**数值**，将另一个操作数转换为数值，然后执行数值比较



（4）如果一个操作数是**对象**，则调用对象的valueOf()方法，再按照上面的规则，如没有valueOf()方法，则调用**Number()**或者**toString()**方法

（5）如一个操作数是**布尔值**，则先转换为数值，再进行比较，**true**转换为**1**，**flase**转换为**0**

7.相等操作符

关系操作符：== !== === !===

（1）如果有一个操作数是**布尔值**，比较前先转换为数值，false转为0，true转为1

（2）如果一个操作数是**字符串**，另一个操作数是**数值**，比较前先将字符串转换为数值

（3）如果一个操作数是对象，另一个操作数不是，则调用对象的valueOf()方法，

（4）null==undefined

（5）在比较相等性前，不能将null和undefined转换为其他任何值

（6）如果一个操作数是NaN，相等操作返回false，不相等操作返回true 。 NaN不等于NaN

（7）如果两个操作数是对象，则比较它们是不是同一个对象，如果两个操作数都指向同一个对象，则相等操作返回true

语句中存在的隐式类型转换

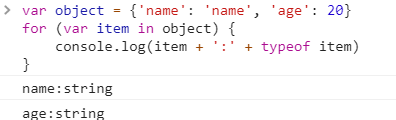
1.if语句 while语句

if（condition）statement1 else statement2

ECMAScript会自动调用Boolean()转换函数将这个表达式

2.for in时的类型转换

**定义对象字面量**时发生从标识符到字符串的隐式转换。

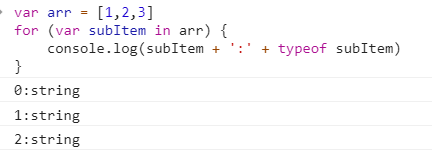
Var object = {‘name’ : ‘name’, ‘age’: 20}

for (var item in object) {

console.log(item + ‘:’ typeof item)

}

**数组的索引**其实也是字符串类型

Vararr = [1, 2, 3]

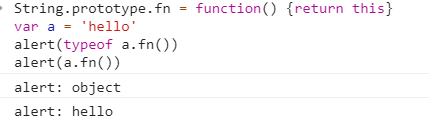
for (var subitem in arr) {

console.log(subitem + ‘:’ typeof subitem)

}

3.alert时存在的隐式类型转换

String类型

String.prototype.fn = function () { return this}

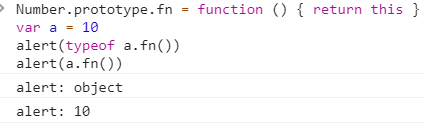
var a = ‘hello’

alert(typeof a.fn())

alert(a.fn())

Number类型

Number.prototype.fn = function () { return this}

var a = 10

alert(typeof a.fn())

alert(a.fn())