对象强制转换为原始值

**一。转换场景**

**1.对象 转 boolean的场景**

**var** a = **new** Boolean(**false**);

**if**(a){

*//do something*

}

**2.对象转 数字的场景**

**var** a = {

i:10,

valueOf: **function**(){**return** **this**.i;},

toString: **function**(){**return** **this**.i\* 2}

};

alert(+a);

alert(''+a); // 先转换为数字，再转为字符串，然后再合并

alert(Number(a));

alert(a>10);

if(a>’13’);

**3.对象转字符串的场景**

**var** a = {

i:10,

valueOf: **function**(){**return** **this**.i;},

toString: **function**(){**return** **this**.i\* 2}

};

alert(String(a));

document.getElementById("pad").innerText = a;

注：对象应用在 + == 或比较运算符的时候，基本上都是转换为数字，即先使用valueOf, 而日期类型除外。（详见<<javascript权威指南第六版>> P54）

**var** b = **new** Date();

*//输出：Thu Sep 07 2017 09:28:59 GMT+0800 (中国标准时间)*

document.getElementById("pad").innerText = b;

**var** b = **new** Date();

*//输出：Thu Sep 07 2017 09:31:18 GMT+0800 (中国标准时间)*

console.log("" + b);

typeof(b+1) //string

typeof(b-1) //number //因为减号不适用“注“中提到的规则

b == b.toString() //true 隐式和显式转换为字符串

b > (b-1) //true 将日期转换为数字

**二. 转换步骤**

**1. 对象转为数字**

a. 如果对象具有valueof()方法，如果该方法返回一个原始值，则js将这个原始值转换为数字，并返回这个数字。

b. 否则，如果对象具有toString()方法，如果后者返回一个原始值，则js将该值转换为数字，并返回。

c. 否则，抛出异常。

**2. 对象转为String**

a. 如果对象具有toString()方法，如果后者返回一个原始值，将原始值转换为字符串，并返回

b. 否则，如果对象具有valueof()方法，如果后者返回一个原始值，将原始值转换为字符串，并返回

c. 否则，抛出异常。

**三. 总结**

因而，在处理对象转换为原始值时，处理的顺序应该是：

1.判断当前环境是要转换成什么类型（数字，布尔型，还是字符串）

2.根据规则二进行转换。

**四. 附录**

**1.valueof()的返回值**

|  |  |
| --- | --- |
| 对象 | 返回值 |
| Array | 返回本身，因为数组也是对象。 |
| Boolean | Boolean 值。 |
| Date | 存储的时间是从 1970 年 1 月 1 日午夜开始计的毫秒数 UTC。 |
| Function | 函数本身。 |
| Number | 数字值。 |
| Object | 对象本身。这是默认情况。 |
| String | 字符串值。 |

Math 和 Error 对象没有 valueOf 方法。  
  
基本上，所有JS数据类型都拥有valueOf和toString这两个方法，null除外。它们俩解决javascript值运算与显示的问题。

**2.toString()的返回值（详见javaScript类型转换表，已经打印出来了）**

**四. 例题**

**1.数组**

**var** a = [];

console.log(+a); *// 0*

结果：输出0

分析：

① 分析 console.log(+a); 该语句应该是将对象转换为数字。

② 首先使用valueof（），而数组是对象，对象的valueof()方法返回了对象本身，而不是原始值，因而要进入下一步，使用toString();

③ 使用toString()方法，根据转换表，空数组将返回””

④ 根据转换表，“”字符串转换为数字为0，

所以，最终结果为0

**2.自定义对象1**

**var** a = {

x: 10,

vauleOf:**function**(){

**return** {

m:12

};

},

toString: **function**(){

**return** “14”;

}

};

console.log(+a);

结果：14

分析：

① 分析 console.log(+a); 该语句应该是将对象转换为数字。

② 首先使用valueof（），该对象重写了valueOf方法，调用它返回一个对象，因而需要继续使用toString()方法

③ 该对象也重写了toString()方法，使用toString()方法，返回”14”

④ 根据转换表，“14”字符串转换为数字为14，

所以，最终结果为14

**2.自定义对象2**

**var** a = {

x: 10,

valueOf:**function**(){

**return** 12;

},

toString: **function**(){

**return** "14";

}

};

console.log(a=="14"); //false

结果：false

分析：

① 分析 console.log(a == “14”); 该语句应该是将对象转换为数字。(参照转换场景的注)

② 首先使用valueof（），返回12

③ 也就是变成了 12 == “14”

④ 强制转换为 “12” == “14”

所以，最终结果为false