## 避免对混合类型使用==运算符

对于以下表达式，结果会是什么呢？

var bol = 1 == {valueOf:function(){return true;}}; // true

结果是true。

这就表明在执行==运算符的时候会出现隐式的类型转换。

比如：

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

if(form.month.value == (today.getMonth() + 1) &&

form.day.value == today.getDate()){

// happy birthday!!

// ...

}

但是事实上，他只是显示的使用Number函数或者一元运算符+将值转换为数字。

为了我让程序更加清晰，我们可以这样写，来凸显一下上面的隐式转换

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

if(+form.month.value == (today.getMonth() + 1) &&

+form.day.value == today.getDate()){

// happy birthday!!

}

加号或者换成Number()我们就能更好的看出来使怎么进行转化的了。

如果不希望有任何的类型转换，那么请使用严格相等===运算符。

以下是我总结的==运算符的隐式转换规则：

如果前者是null，后者是undefined：

不进行任何转换，直接判定为true；（因为undefined是null的派生）。

如果前者是undefined，后者是undefined：

基本原理同上，前后者调换。

如果前者是原始类型string/number/boolean，后者也是string/number/boolean：

优先尝试对两边都转换成数字，比较两个数字是否相等。如果两边都无法转换为数字，则直接进行比较（基本上是转换不了数字的字符串或者是其他无法转换成数字的，比如：NaN、正负Infinity等。。。）。

如果前者是原始类型string/number/boolean，后者是对象（非Date对象）

找出后者的结果与前者进行判断，之后的两者判断原理同上。

后者如何得到结果：具体的判断过程依旧为传统对象隐式转换老方式，先对对象调用valueof，无法得到可以和前者进行比较的结果就再次调用toString得到结果在与前者进行判断。如果都不能，则报错。

如果前者是对象（非Date对象），后者是原始类型string/number/boolean

基本原理同上，前后者调换。

如果前者是对象（非Date对象），后者也是对象（非Date对象）。

两者都进行上面所述老方式转换，之后进行比较。

如果对象有Date对象。

Date对象比较特殊，他的“转换老方式”和其他对象不同，是先调用toString之后调用valueOf。不知道为什么，ECMAScript就是这么规定的。

还有一个可能头昏脑涨的时候犯的一个低级错误。

var date = **new** *Date*("2018/12/31");

var bol = date == "2018/12/31"; // false

在这里date明显是一个Date对象，是一个对象，本身并不是一个经过自定义整理具备一定时间格式的字符串。虽然我们在输出的Date对象的时候输出一个非常全且非常长的字符串的时间格式。这是因为js在我们输出Date对像的时候默认调用了他的toString方法。

值得注意的是，在console打印字符串String对象的时候没自动调用toString方法和valueOf方法。但是如果alert或者write String对象，还是会输出字符串的,说明调用了toString或者ValueOf方法，**但是具体哪个需要考证**。

var em = {

name : "方向明",

age:21,

valueOf:function(){

return "valueof";

},

toString:function(){

return "toString";

}

};

var a = **new** *String*("asd");

var date = **new** *Date*("2018/12/31");

var bol = date == "2018/12/31"; // false

*console*.log(typeof date); // object

*console*.log(date); // Mon Dec 31 2018 00:00:00 GMT+0800 (中国标准时间)

*console*.log(a); // String {"asd"}

*console*.log(em) // {name: "方向明", age: 21, valueOf: ƒ, toString: ƒ}

因此当我们需要进行比较时间的时候，可以先对date保存的时间对象信息进行整理，提取出我们想要的部分拼接为想要的字符串形式。

比如想要实现前面的时间的==比较；

function toYMD(*date*){

var year = date.getFullYear(),

month = date.getMonth() + 1; // 因为月份是从0开始的

day = date.getDate();

return year

+ "/" + (month < 10 ? '0' + month : month)

+ "/" + (day < 10 ? '0' + day : day);

}

var date = **new** *Date*("2000/1/1");

*console*.log(toYMD(date)); // 2000/01/01

这样输出的格式就能够比较了。