# 每日一题 - 如何令a == 1 && a == 2 && a == 3 返回true？

### 信息卡片

* 时间：2019-08-26
* tag：开放问题 值比较

### 问题描述

在 JavaScript 中， (a == 1 && a == 2 && a == 3) 是否有可能为 true ？

### 参考实现

参考解决思路：a是一个对象或函数，每次调用取值都不一样，以有序的规律变化就能实现多等

###### 方案一

使用getter存储器

var temp = 1;  
Object.defineProperty(window, 'a', {  
 get: function() { // 每次取值，temp+1  
 return this.temp++  
 }  
});  
(a == 1 && a == 2 && a == 3); // true  
(a === 1 && a === 2 && a === 3); // true

这个是使用getter存储器的方式，也就是以全局变量temp存储一个值，每次调用的时候都++1使得调用a每次都递增1

###### 方案二

重写valueOf() / toString()

var a = {  
 value: 1,  
 valueOf: function() {  
 return this.value++;  
 }  
}  
(a == 1 && a == 2 && a == 3); // true

**方案说明**

从表面看，应该是valueOf()每次都被调用了，但是为什么会这样？我们又没有调用它。

?> 这里的valueOf为什么会被调用？

* 原因参考==转换规则

1. 如果一个是null，一个是undefined，则它们相等
2. 如果一个是数字，一个是字符串，先将字符串转换成数字，然后使用转换后的值进行比较
3. 如果其中的一个值为true，则转换成1再进行比较；如果其中一个值为false，这转换成0再进行比较
4. 如果一个值是对象，另一个值是数字或者字符串，则将对象转换成原始值再进行比较。转换成字符串时，会先调用toString()，如果没有toString()方法或者返回的不是一个原始值，则再调用valueOf()，如果还是不存在或者返回不是原始值，则会抛出一个类型错误的异常。返回的原始值会被转换成字符串；如果转换成数字时，也是类似的，不过是会先调用valueOf()，再调用toString()，返回的原始值会被转换成数字
5. 其他不同类型之间的比较均不相等

所以在这里使用a与这些字符进行比较时会被转换成数字，此时会默认调用字符串的valueOf()方法，我们将这个方法进行重写，用于拦截处理a的值

* 同理可以使用toString方法处理，因为字符串转数字类型时会涉及到valueOf()和toString()，道理一样

let a = {  
 value: 1,  
 toString: function () {  
 return a.value++; // 这里为什么不用this而已a？因为this作用域可变  
 }  
}  
console.log(a == 1 && a == 2 && a == 3); // true

只要符合递增规则的，a就可以实现多等，因为此a非彼a

###### 方案三

ES6 Proxy

var a = new Proxy({ i: 0 }, {  
 get: (target, name) => name === Symbol.toPrimitive ? () => ++target.i : target[name],  
});  
console.log(a == 1 && a == 2 && a == 3); // true

###### 方案四

数字变量名

var a = 1;  
var ﾠ1 = a;  
var ﾠ2 = a;  
var ﾠ3 = a;  
console.log( a ==ﾠ1 && a ==ﾠ2 && a ==ﾠ3 );

###### 方案五：join + shift

* 对于对象数组进行比较时，这里数组a每次比较的时候都会默认调用toString()，然后toString()又会默认调用join()，这里将join()改为shift()，意思是删除第一个数组元素值并返回
* 所以这样调用每次都会导致a数组删除第一个值并且返回删除掉的那个值，结合这样的规律，每次比较都取出对应位置的值
* 这里是1、2、3，只要符合规律返回的值就行

var a =[1,2,3];  
a.join = a.shift;  
console.log(a); // (3) [1, 2, 3, join: ƒ]  
// console.log( a ==ﾠ1 && a ==ﾠ2 && a ==ﾠ3 ); // true  
console.log(a == 1); // true  
console.log(a); // (2) [2, 3, join: ƒ]  
console.log(a == 2); // true  
console.log(a); // [3, join: ƒ]  
console.log(a == 3); // true  
console.log(a); // [join: ƒ]

### 扩展

* 由上面我们可以总结出一个规律，只要符合一个递增规律的，我们就可以实现对象多等
* 同理我们可以扩展到更多更复杂规律的对象比较中

### 参考文章

[关于如何使(a === 1 && a === 2 && a === 3)返回true问题的思考](https://github.com/azl397985856/fe-interview/issues/cnblogs.com/shapeY/p/10183749.html) | [js 中怎么使 if(aﾠ==1 && a== 2 && ﾠa==3) 返回 true？](https://juejin.im/post/5c219e92e51d450d5a01aca7)