"==="叫做严格运算符，"=="叫做相等运算符。

严格运算符的运算规则如下，
(1)不同类型值
如果两个值的类型不同，直接返回false。
(2)同一类的原始类型值

同一类型的原始类型的值（数值、字符串、布尔值）比较时，值相同就返回true，值不同就返回false。

(3)同一类的复合类型值

两个复合类型（对象、数组、函数）的数据比较时，不是比较它们的值是否相等，而是比较它们是否指向同一个对象。

(4)undefined和null

undefined 和 null 与自身严格相等。

```
null === null  //true
undefined === undefined  //true
```

相等运算符在比较相同类型的数据时，与严格相等运算符完全一样。

在比较不同类型的数据时，相等运算符会先将数据进行类型转换，然后再用严格相等运算符比较。类型转换规则如下：

(1)原始类型的值

原始类型的数据会转换成数值类型再进行比较。**字符串和布尔值都会转换成数值，所以题主的问题中会有第二个string输出。**

(2)对象与原始类型值比较

对象（这里指广义的对象，包括数值和函数）与原始类型的值比较时，对象转化成原始类型的值，再进行比较。

(3)undefined和null

undefined和null与其他类型的值比较时，结果都为false，它们互相比较时结果为true。

(4)相等运算符的缺点

相等运算符隐藏的类型转换，会带来一些违反直觉的结果。

```
'' == '0'           // false
0 == ''             // true
0 == '0'            // true

false == 'false'    // false
```

```
false == '0'         // true

false == undefined   // false
false == null        // false
null == undefined    // true

' \t\r\n ' == 0      // true
```

这就是为什么**建议尽量不要使用相等运算符。**

至于使用相等运算符会不会对后续代码造成意外影响，答案是有可能会。

```
var a = undefined;
if(!a){
    console.log("1"); //1
}
```

```
var a = undefined;
if(a == null){
    console.log("1"); //1
}
```

```
var a = undefined;
if(a === null){
    console.log("1"); //无输出
}
```

也就是说当a为undefined时，输出的值会有变化，而在编程中对象变成undefined实在是太常见了。