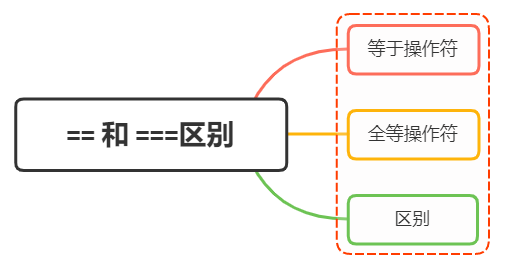
# 面试官：== 和 ===区别，分别在什么情况使用



## 一、等于操作符

等于操作符用两个等于号（ == ）表示，如果操作数相等，则会返回 true

前面文章，我们提到在JavaScript中存在隐式转换。等于操作符（==）在比较中会先进行类型转换，再确定操作数是否相等

遵循以下规则：

如果任一操作数是布尔值，则将其转换为数值再比较是否相等

let result1 = (true == 1); // true

如果一个操作数是字符串，另一个操作数是数值，则尝试将字符串转换为数值，再比较是否相等

let result1 = ("55" == 55); // true

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()方法取得其原始值，再根据前面的规则进行比较

let obj = {valueOf:function(){return 1}}  
let result1 = (obj == 1); // true

null和undefined相等

let result1 = (null == undefined ); // true

如果有任一操作数是 NaN ，则相等操作符返回 false

let result1 = (NaN == NaN ); // false

如果两个操作数都是对象，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，则相等操作符返回true

let obj1 = {name:"xxx"}  
let obj2 = {name:"xxx"}  
let result1 = (obj1 == obj2 ); // false

下面进一步做个小结：

* 两个都为简单类型，字符串和布尔值都会转换成数值，再比较
* 简单类型与引用类型比较，对象转化成其原始类型的值，再比较
* 两个都为引用类型，则比较它们是否指向同一个对象
* null 和 undefined 相等
* 存在 NaN 则返回 false

## 二、全等操作符

全等操作符由 3 个等于号（ === ）表示，只有两个操作数在不转换的前提下相等才返回 true。即类型相同，值也需相同

let result1 = ("55" === 55); // false，不相等，因为数据类型不同  
let result2 = (55 === 55); // true，相等，因为数据类型相同值也相同

undefined 和 null 与自身严格相等

let result1 = (null === null) //true  
let result2 = (undefined === undefined) //true

## 三、区别

相等操作符（==）会做类型转换，再进行值的比较，全等运算符不会做类型转换

let result1 = ("55" === 55); // false，不相等，因为数据类型不同  
let result2 = (55 === 55); // true，相等，因为数据类型相同值也相同

null 和 undefined 比较，相等操作符（==）为true，全等为false

let result1 = (null == undefined ); // true  
let result2 = (null === undefined); // false

### 小结

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

但在比较null的情况的时候，我们一般使用相等操作符==

const obj = {};  
  
if(obj.x == null){  
 console.log("1"); //执行  
}

等同于下面写法

if(obj.x === null || obj.x === undefined) {  
 ...  
}

使用相等操作符（==）的写法明显更加简洁了

所以，除了在比较对象属性为null或者undefined的情况下，我们可以使用相等操作符（==），其他情况建议一律使用全等操作符（===）