**关系运算符执行的是比较运算。每个关系运算符都返回一个布尔值。**

关系运算符小于、大于、小于等于和大于等于执行的是两个数的比较运算，比较方式与算术比较运算相同。

比较规则：

1. 两个数字比较，按数字实际大小比较，返回比较结果（true、false）
2. 数字和字母比较，字母转成数字（parseInt），返回比较结果（true、false）
3. 两个字母比较，运算符两端的字母按顺序逐个比较字母的字节码（区分大小写），返回比较结果（true、false）
4. NaN 与任何比较都返回false
5. Infinity和-Infinity 同数字比较时，按实际大小返回结果（true、false）

**等性运算符==、===**

等号和非等号

在 ECMAScript 中，等号由双等号（==）表示，当且仅当两个运算数相等时，它返回 true。非等号由感叹号加等号（!=）表示，当且仅当两个运算数不相等时，它返回 true。为确定两个运算数是否相等，这两个运算符都会进行类型转换。

执行类型转换的规则如下：

*如果一个运算数是 Boolean 值，在检查相等性之前，把它转换成数字值。false 转换成 0，true 为 1。*

*如果一个运算数是字符串，另一个是数字，在检查相等性之前，要尝试把字符串转换成数字。*

*如果一个运算数是对象，另一个是字符串，在检查相等性之前，要尝试把对象转换成字符串。*

*如果一个运算数是对象，另一个是数字，在检查相等性之前，要尝试把对象转换成数字。*

在比较时，该运算符还遵守下列规则：

*值 null 和 undefined 相等。*

*在检查相等性时，不能把 null 和 undefined 转换成其他值。*

*如果某个运算数是 NaN，等号将返回 false，非等号将返回 true。*

*如果两个运算数都是对象，那么比较的是它们的引用值。如果两个运算数指向同一对象，那么等号返回 true，否则两个运算数不等。*

**重要提示：即使两个数都是 NaN，等号仍然返回 false，因为根据规则，NaN 不等于 NaN。**

下表列出了一些特殊情况，以及它们的结果：

表达式 值

null == undefined true

"NaN" == NaN false

5 == NaN false

NaN == NaN false

NaN != NaN true

false == 0 true

true == 1 true

true == 2 false

undefined == 0 false

null == 0 false

"5" == 5 true

**全等号和非全等号**

等号和非等号的同类运算符是全等号和非全等号。这两个运算符所做的与等号和非等号相同，只是它们在检查相等性前，不执行类型转换。

全等号由三个等号表示（===），只有在无需类型转换运算数就相等的情况下，才返回 true。

例如：

var sNum = "66";

var iNum = 66;

alert(sNum == iNum); //输出 "true"

alert(sNum === iNum); //输出 "false"

在这段代码中，第一个 alert 使用等号来比较字符串 "66" 和数字 66，输出 "true"。如前所述，这是因为字符串 "66" 将被转换成数字 66，，然后才与另一个数字 66 进行比较。第二个 alert 使用全等号在没有类型转换的情况下比较字符串和数字，当然，字符串不等于数字，所以输出 "false"。

非全等号由感叹号加两个等号（!==）表示，只有在无需类型转换运算数不相等的情况下，才返回 true。

例如：

var sNum = "66";

var iNum = 66;

alert(sNum != iNum); //输出 "false"

alert(sNum !== iNum); //输出 "true"

这里，第一个 alert 使用非等号，把字符串 "66" 转换成数字 66，使得它与第二个运算数 66 相等。因此，计算结果为 "false"，因为两个运算数是相等的。第二个 alert 使用的非全等号。该运算是在问："sNum" 与 "iNum" 不同吗？这个问题的答案是：是的（true），因为 sNum 是字符串，而 iNum 是数字，它们当然不同。