布尔运算符

目录

- 概述
- 取反运算符（!）
- 且运算符（&&）
- 或运算符（||）
- 三元条件运算符（?:）

概述

布尔运算符用于将表达式转为布尔值，一共包含四个运算符。

- 取反运算符：`!`
- 且运算符：`&&`
- 或运算符：`||`
- 三元运算符：`?:`

取反运算符（!）

取反运算符是一个感叹号，用于将布尔值变为相反值，即 `true` 变成 `false`，`false` 变成 `true`。

```
!true // false
!false // true
```

对于非布尔值，取反运算符会将其转为布尔值。可以这样记忆，以下六个值取反后为 `true`，其他值都为 `false`。

- `undefined`
- `null`
- `false`
- `0`
- `NaN`

- 空字符串（`''`）

```
!undefined // true
!null // true
!0 // true
!NaN // true
!"" // true

!54 // false
!'hello' // false
![] // false
!{} // false
```

上面代码中，不管什么类型的值，经过取反运算后，都变成了布尔值。

如果对一个值连续做两次取反运算，等于将其转为对应的布尔值，与`Boolean`函数的作用相同。这是一种常用的类型转换的写法。

```
!!x
// 等同于
Boolean(x)
```

上面代码中，不管`x`是什么类型的值，经过两次取反运算后，变成了与`Boolean`函数结果相同的布尔值。所以，两次取反就是将一个值转为布尔值的简便写法。

且运算符（&&）

且运算符（`&&`）往往用于多个表达式的求值。

它的运算规则是：如果第一个运算子的布尔值为`true`，则返回第二个运算子的值（注意是值，不是布尔值）；如果第一个运算子的布尔值为`false`，则直接返回第一个运算子的值，且不再对第二个运算子求值。

```
't' && '' // ""
't' && 'f' // "f"
't' && (1 + 2) // 3
'' && 'f' // ""
'' && '' // ""

var x = 1;
```

```
(1 - 1) && ( x += 1) // 0
x // 1
```

上面代码的最后一个例子，由于且运算符的第一个运算子的布尔值为 `false`，则直接返回它的值 `0`，而不再对第二个运算子求值，所以变量 `x` 的值没变。

这种跳过第二个运算子的机制，被称为“短路”。有些程序员喜欢用它取代 `if` 结构，比如下面是一段 `if` 结构的代码，就可以用且运算符改写。

```
if (i) {
  doSomething();
}

// 等价于

i && doSomething();
```

上面代码的两种写法是等价的，但是后一种不容易看出目的，也不容易除错，建议谨慎使用。

且运算符可以多个连用，这时返回第一个布尔值为 `false` 的表达式的值。如果所有表达式的布尔值都为 `true`，则返回最后一个表达式的值。

```
true && 'foo' && '' && 4 && 'foo' && true
// ''

1 && 2 && 3
// 3
```

上面代码中，例一里面，第一个布尔值为 `false` 的表达式为第三个表达式，所以得到一个空字符串。例二里面，所有表达式的布尔值都是 `true`，所以返回最后一个表达式的值 `3`。

或运算符（||）

或运算符（`||`）也用于多个表达式的求值。它的运算规则是：如果第一个运算子的布尔值为 `true`，则返回第一个运算子的值，且不再对第二个运算子求值；如果第一个运算子的布尔值为 `false`，则返回第二个运算子的值。

```
't' || '' // "t"
't' || 'f' // "t"
'' || 'f' // "f"
```

```
'' || '' // ""
```

短路规则对这个运算符也适用。

```
var x = 1;
true || (x = 2) // true
x // 1
```

上面代码中，或运算符的第一个运算子为 `true`，所以直接返回 `true`，不再运行第二个运算子。所以，`x` 的值没有改变。这种只通过第一个表达式的值，控制是否运行第二个表达式的机制，就称为“短路”（short-cut）。

或运算符可以多个连用，这时返回第一个布尔值为 `true` 的表达式的值。如果所有表达式都为 `false`，则返回最后一个表达式的值。

```
false || 0 || '' || 4 || 'foo' || true
// 4

false || 0 || ''
// ''
```

上面代码中，例一里面，第一个布尔值为 `true` 的表达式是第四个表达式，所以得到数值 4。例二里面，所有表达式的布尔值都为 `false`，所以返回最后一个表达式的值。

或运算符常用于为一个变量设置默认值。

```
function saveText(text) {
  text = text || '';
  // ...
}

// 或者写成
saveText(this.text || '')
```

上面代码表示，如果函数调用时，没有提供参数，则该参数默认设置为空字符串。

三元条件运算符（?:）

三元条件运算符由问号（?）和冒号（:）组成，分隔三个表达式。它是 JavaScript 语言唯一一个需要三个运算子的运算符。如果第一个表达式的布尔值为 `true`，则返回第二个表达式的值，否则返回第三个表达式的值。

```
't' ? 'hello' : 'world' // "hello"
0 ? 'hello' : 'world' // "world"
```

上面代码的 `t` 和 `0` 的布尔值分别为 `true` 和 `false`，所以分别返回第二个和第三个表达式的值。

通常来说，三元条件表达式与 `if...else` 语句具有同样表达效果，前者可以表达的，后者也能表达。但是两者具有一个重大差别，`if...else` 是语句，没有返回值；三元条件表达式是表达式，具有返回值。所以，在需要返回值的场合，只能使用三元条件表达式，而不能使用 `if..else`。

```
console.log(true ? 'T' : 'F');
```

上面代码中，`console.log` 方法的参数必须是一个表达式，这时就只能使用三元条件表达式。如果要用 `if...else` 语句，就必须改变整个代码写法了。

版本号 #2

由 壳壳虫 创建于 2018-01-09 14:23:47 UTC

由 壳壳虫 更新于 2020-05-14 16:12:24 UTC