

# JavaScript 运算符









#### ◆ 运算符

- ◆ 算数运算符
- ◆ 递增和递减运算符
- ◆ 比较运算符
- ◆ 逻辑运算符
- ◆ 赋值运算符
- ◆ 运算符优先级

# 1. 运算符



运算符 (operator) 也被称为<mark>操作符</mark>,是用于实现赋值、比较和执行算数运算等功能的符号。

#### JavaScript中常用的运算符有:

- 算数运算符
- 递增和递减运算符
- 比较运算符
- 逻辑运算符
- 赋值运算符





- ◆ 运算符
- ◆ 算数运算符
- ◆ 递增和递减运算符
- ◆ 比较运算符
- ◆ 逻辑运算符
- ◆ 赋值运算符
- ◆ 运算符优先级



#### 2.1 算术运算符概述

概念: 算术运算使用的符号, 用于执行两个变量或值的算术运算。

运算符	描述	实例	
+	מל	10 + 20 = 30	
-	减	10 - 20 = -10	
*	乘	10 * 20 = 200	
/	除	10 / 20 = 0.5	
%	取余数(取模)	返回除法的余数 9 % 2 = 1	



#### 2.2 浮点数的精度问题

浮点数值的最高精度是 17 位小数, 但在进行算术计算时其精确度远远不如整数。

所以:不要直接判断两个浮点数是否相等!



#### 2.3 课堂提问

1. 我们怎么判断 一个数能够被整除呢?

它的余数是0 就说明这个数能被整除,这就是% 取余运算符的主要用途

2. 请问 1 + 2 \* 3 结果是?

结果是7,注意算术运算符优先级的,先乘除,后加减,有小括号先算小括号里面的



#### 2.4 表达式和返回值

表达式:是由数字、运算符、变量等以能求得数值的有意义排列方法所得的组合

简单理解: 是由数字、运算符、变量等组成的式子

表达式最终都会有一个结果,返回给我们,我们成为返回值





- ◆ 运算符
- ◆ 算数运算符
- ◆ 递增和递减运算符
- ◆ 比较运算符
- ◆ 逻辑运算符
- ◆ 赋值运算符
- ◆ 运算符优先级



#### 3.1 递增和递减运算符概述

如果需要反复给数字变量添加或减去1,可以使用递增(++)和递减(--)运算符来完成。

在 JavaScript 中,递增(++)和递减(--)既可以放在变量前面,也可以放在变量后面。<mark>放在变量前面</mark>时, 我们可以称为前置递增(递减)运算符,放在变量后面时,我们可以称为后置递增(递减)运算符。

注意: 递增和递减运算符必须和变量配合使用。



#### 3.2 递增运算符

#### 1. 前置递增运算符

```
++num 前置递增, 就是自加1, 类似于 num = num + 1, 但是 ++num 写起来更简单。
```

使用口诀: 先自加, 后返回值

```
var num = 10;
alert(++num + 10);  // 21
```



#### 3.2 递增运算符

#### 2. 后置递增运算符

```
num++ 后置递增, 就是自加1, 类似于 num = num + 1, 但是 num++ 写起来更简单。
```

使用口诀: 先返回原值, 后自加

```
var num = 10;
alert(10 + num++); // 20
```





```
var a = 10;
++a;
var b = ++a + 2;
console.log(b);
var c = 10;
C++;
var d = c++ + 2;
console.log(d);
var e = 10;
var f = e++ + ++e;
console.log(f);
```



#### 3.3 前置递增和后置递增小结

- 前置递增和后置递增运算符可以简化代码的编写,让变量的值 + 1 比以前写法更简单
- 单独使用时,运行结果相同
- 与其他代码联用时,执行结果会不同
- 后置: 先原值运算, 后自加 (先人后己)
- 前置: 先自加, 后运算(先已后人)
- 开发时,大多使用后置递增/减,并且代码独占一行,例如: num++; 或者 num--;





- ◆ 运算符
- ◆ 算数运算符
- ◆ 递增和递减运算符
- ◆ 比较运算符
- ◆ 逻辑运算符
- ◆ 赋值运算符
- ◆ 运算符优先级

### 4. 比较运算符



#### 4.1 比较运算符概述

概念:比较运算符(关系运算符)是<mark>两个数据进行比较时所使用的运算符</mark>,比较运算后,会<mark>返回一个布尔值</mark> (true / false)作为比较运算的结果。

运算符名称	说明	案例	结果
<	小于号	1 < 2	true
>	大于号	1 > 2	false
>=	大于等于号 (大于或者等于)	2 >= 2	true
<=	小于等于号 (小于或者等于)	3 <= 2	false
==	判等号 (会转型)	37 == 37	true
<u>!</u> =	不等号	37 != 37	false
=== !==	全等 要求值和 数据类型都一致	37 === '37'	false

# 4. 比较运算符



#### 4.2 =小结

符号	作用	用法	
=	赋值	把右边给左边	
==	判断	判断两边值是否相等 (注意此时有隐式转换)	
===	全等	判断两边的值和数据类型是否完全相同	

```
console.log(18 == '18');
console.log(18 === '18');
```

# 4. 比较运算符



### ❷ 课堂练习

```
var num1 = 10;
var num2 = 100;
var res1 = num1 > num2;
var res2 = num1 == 11;
var res3 = num1 != num2;
```





- ◆ 运算符
- ◆ 算数运算符
- ◆ 递增和递减运算符
- ◆ 比较运算符
- ◆ 逻辑运算符
- ◆ 赋值运算符
- ◆ 运算符优先级



#### 5.1 逻辑运算符概述

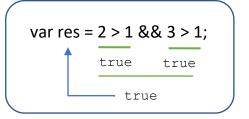
概念:逻辑运算符是用来进行布尔值运算的运算符,其返回值也是布尔值。后面开发中经常用于多个条件的判断

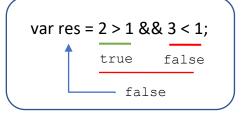
逻辑运算符	说明	案例
&&	"逻辑与",简称 "与" and	true && false
П	"逻辑或",简称 "或" or	true    false
!	"逻辑非",简称 "非" not	! true



#### 5.2 逻辑与&&

两边都是 true才返回 true, 否则返回 false

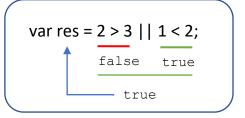


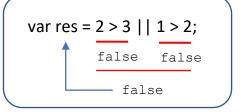




#### 5.3 逻辑或 ||

两边都为 false 才返回 false, 否则都为true







#### 5.3 逻辑非!

逻辑非(!)也叫作取反符,用来取一个布尔值相反的值,如 true 的相反值是 false

```
var isOk = !true;
console.log(isOk); // false
```





```
var num = 7;
var str = "我爱你~中国~";
console.log(num > 5 && str.length >= num);

console.log(num < 5 && str.length >= num);

console.log(!(num < 10));

console.log(!(num < 10 || str.length == num));</pre>
```



#### 5.4 短路运算 (逻辑中断)

短路运算的原理: 当有多个表达式 (值) 时,左边的表达式值可以确定结果时,就不再继续运算右边的表达式的值;

#### 1. 逻辑与

- 语法: 表达式1 && 表达式2
- 如果第一个表达式的值为真,则返回表达式2
- 如果第一个表达式的值为假,则返回表达式1

```
console.log( 123 && 456 ); // 456
console.log( 0 && 456 ); // 0
console.log( 123 && 456&& 789 ); // 789
```



#### 5.4 逻辑中断 (短路操作)

#### 2. 逻辑或

- 语法: 表达式1 || 表达式2
- 如果第一个表达式的值为真,则返回表达式1
- 如果第一个表达式的值为假,则返回表达式2

```
console.log( 123 || 456 );  // 123
console.log( 0 || 456 );  // 456
console.log( 123 || 456 || 789 );  // 123
```



#### 5.4 逻辑中断 (短路操作)

```
var num = 0;
console.log(123 || num++);
console.log(num);
```





- ◆ 运算符
- ◆ 算数运算符
- ◆ 递增和递减运算符
- ◆ 比较运算符
- ◆ 逻辑运算符
- ◆ 赋值运算符
- ◆ 运算符优先级

# 6. 赋值运算符



概念: 用来把数据赋值给变量的运算符。

赋值运算符	说明	案例
=	直接赋值	var usrName = '我是值';
+=, -=	加、减一个数后在赋值	var age = 10; age+=5; // 15
*=、/=、%=	乘、除、取模 后在赋值	var age = 2; age*=5; // 10

```
var age = 10;
age += 5;  // 相当于 age = age + 5;
age -= 5;  // 相当于 age = age - 5;
age *= 10; // 相当于 age = age * 10;
```





- ◆ 运算符
- ◆ 算数运算符
- ◆ 递增和递减运算符
- ◆ 比较运算符
- ◆ 逻辑运算符
- ◆ 赋值运算符
- ◆ 运算符优先级

# 7. 运算符优先级



优先级	运算符	顺序
1	小括号	()
2	一元运算符	++ !
3	算数运算符	先*/% 后+-
4	关系运算符	> >= < <=
5	相等运算符	== [== !==
6	逻辑运算符	先 & & 后
7	赋值运算符	=
8	逗号运算符	,

- 一元运算符里面的逻辑非优先级很高
- 逻辑与比逻辑或优先级高

### 7. 运算符优先级



#### 练习 1

```
console.log( 4 >= 6 || '人' != '阿凡达' && !(12 * 2 == 144) && true)
var num = 10;
console.log( 5 == num / 2 && (2 + 2 * num).toString() === '22');
```

### 7. 运算符优先级



#### 练习 2

```
var a = 3 > 5 && 2 < 7 && 3 == 4;
console.log(a);
var b = 3 \le 4 \mid \mid 3 > 1 \mid \mid 3 \mid = 2;
console.log(b);
var c = 2 === "2";
console.log(c);
var d = !c || b && a ;
console.log(d);
```



传智播客旗下高端IT教育品牌