第七课 运算符

学习目录

- **算术运算符**
- **一元运算符**
- **比较运算符**
- **逻辑运算符**
- **赋值运算符**
- **其他运算符（位运算符、字符串运算符）**

js 中定义了一组用来对数据值进行操作的的运算符，他是以某些数据值作为操作数，并返回一定的操作后的数据值。

①算术运算符

算术运算符用于执行两个变量或值的运算，算术运算符以数值（字面量或变量）作为其操作数，并返回一个单个数值。js 中总共有加减乘除模这几类算术运算符。还可以把多种操作组合在一起：1+4*5。

加法运算符的作用是数值求和，或者字符串拼接。

减法运算符使两个操作数相减，结果是它们的差值。

除法运算符的结果是操作数的商 ，左操作数是被除数，右操作数是除数。

乘法运算符的结果是操作数的乘积。

求余运算符返回第一个操作数对第二个操作数的模。

运算符	描述	例子
+	加法	var x = 3 + 2
-	减法	var x = 3 - 2
*	乘法	var x = 3 * 2
/	除法	var x = 3 / 2
%	余数	var x = 9 % 2

②一元运算符

一元运算符只有一个参数，即要操作的数据值。一元加，一元减，前置递增、递减操作符，后置递增、递减操作符。一元加、一元减用法与数学中的用法相同，用来把 js 中任何数据类型转换成数字类型。递增操作符、递减操作符用来把任何数据类型先转换成数字，然后再进行加一或减一的操作。

一元加（得到正数）

```
var num = 20;
num = +num;
```

一元减（得到负数）

```
var num = 20;
num = -num;
```

前置递增、递减（先会对数据值进行加一或减一的操作，在让数据值参与到后面的运算形式也就是在变量前放两个加号或者两个减号）

```
var num1 = 2;

var num2 = 20;

--num 1 + num2;   // 1 + 20
```

后置递增、递减（先让数据值参与运算，在对数据值进行加一或减一的操作）

```
var num1 = 2;

var num2 = 20;

var num3 = num1-- + num2++;//等于 "22"

var num4 = num1 + num2;       //等于 "22"
```

③比较运算符

比较运算符也称为关系运算符，在逻辑语句中使用，以判断变量或值是否相等，比较得到的值通常是布尔值 true 或 false。比较不同类型的数据也许会出现不可预料的结果。基础的比较运算符有小于(<)、大于(>)、小于等于(<=)、大于等于(>=)、等于(==)、不等(!=)、全等(===)、不全等(!==)这几种。

全等(===)值相等并且类型相等。

```
alert('100'===100)//false
```

不全等(!==)值不相等或类型不相等。

```
alert('100'!==100)//true
```

当比较两个字符串时，"2" 大于 "11"，因为（按照字母排序）1 小于 2。

比较运算符可用在条件语句中对值进行比较，并根据结果进行其他操作。

```
var num = 11;
if (num < 60) alert("不及格");
```

在默认情况下 null 和 undefined 是相等的；

在默认情况下，如果两个比较值都是对象 Object 类型，要比较他们是否是指向同一个内存地址的对象，如果都指向同一个对象，则返回 true，否则返回 false。

④逻辑运算符

逻辑运算符用于判定变量或值之间的逻辑。逻辑运算符用三种方式表示逻辑与&&、逻辑或||、逻辑非!，逻辑运算符的结果为布尔类型。

```
var a = 3;
var b = -2;
alert(a > 0 && b > 0);//false
alert(a > 0 || b > 0);//true
alert(!(a > 0 || b > 0));//false
```

逻辑与&&判断结果标准

```
alert(true && true);//true
alert(true && false);//false
alert(false && true);//false
alert(false && false);//false
```

第一个判断条件	第二个判断条件	返回结果
true	true	true
true	false	false
false	true	false
false	false	false

逻辑或||判断结果标准

alert(true || true);//true

alert(true || false);//true

alert(false || true);//true

alert(false || false);//false

第一个判断条件	第二个判断条件	返回结果
true	true	true
true	false	true
false	true	true
false	false	false

逻辑非!判断结果标准

对结果取反，双重非（!!）运算符 n1 = !!true，则 n1 返回 true。

alert(!true);//false

alert(!false);//true

alert!(!!false);//false

null，NaN，0，空字符串，undefined 都会在逻辑判断中被转换成 false。

&&如果第一个表达式为 false，则之后的逻辑判断就不在执行。

||如果第一个表达式为 true,则之后的逻辑判断就不在执行。

⑤赋值运算符

赋值运算符基于右值的值，给左边的变量赋值。

```
var x = 11;
```

把一个值赋给多个变量，可以以链式使用赋值运算符。

```
var y = 0;

var z = 0;

y = z = x;

alert(x);

alert(y);

alert(z);
```

赋值　　x = y　　x = y

加赋值　x += y　x = x + y

减赋值　x -= y　　x = x - y

乘赋值　x *= y　　x = x * y

除赋值　x /= y　　x = x / y

模赋值　x %= y　x = x % y

x += 5;等价于 x = x + 5;相当于 x 变量本身加 100。

什么是表达式？

表达式指的是可用于计算的式子，由运算值和运算符(可选)构成，可能会产生运算结果的语法结构。js 表达式会计算得到返回值，其中，单值表达式的结果是值本身，复合表达式结果是根据运算符进行运算的结果值。

单值表达式（基本表达式）

最简单的表达式是基本表达式，基本表达式包含常量（NaN）或直接量（直接在程序中出现的常数值。123，“helloxingxingketang”）、关键字（if）和变量（var a = 1）。

复合表达式

使用运算符组合各类基本表达式进行运算的语法结构，目前所讲的重点是运算符构成的表达式。

ps:表达式扩展知识

对象和数组初始化表达式

```
var obj = {x:1,y:2}
```

```
var arr = [1,2,3]
```

函数定义表达式

```
var fun = function (x){return x*x}
```

对象实例化表达式

```
var obj = new Object()
```

正则表达式

```
var reg=/^\w+@[a-zA-Z0-9]{2,10}(?:\.[a-z]{2,4}){1,3}$/;
```

⑥其他运算符

位运算符

位运算符是在数字底层（即表示数字的 32 个数位）进行操作的。位运算符是对它的操作数当成 32 位的比特序列（由 0 和 1 组成），从而操作数字的二进制形式，不过返回值依然是标准的 js 数值。

var Num1 = 25; //十进制 25 等于 二进制 00000000000000000000000000011001

var Num2 = ~iNum1; //~按位取反 转换为 11111111111111111111111111100110

alert(Num2); //输出 "-26"

拓展阅读：https://www.w3school.com.cn/js/pro_js_operators_bitwise.asp。

逗号运算符

逗号运算符常用变量声明中。

var Num1 = 1, Num = 2, Num3 = 3;

三元运算符

条件？条件成立执行：条件不成立执行。如果条件为 true ，则整个表达式的结果就是条件成立执行的结果，如果条件为 false，则整个表达式的结果就是条件不成立执行的结果。相当于简单的 if()else()语句。

var num = 3 ;

if(num > 1 && num <= 3){alert(‘数据在 1 和 3 之间’)}else{alert(‘数据不在 1 和 3 之间’)}

运算符优先级

优先级	运算类型	运算符（...表示相关操作数）
由高到低	圆括号	(...)
	成员访问	... .
	需计算的成员访问	... [...]
	new (带参数列表)	new ... (...)
	函数调用	... (...)
	new (无参数列表)	new ...
	后置递增(运算符在后)	... ++
	后置递减(运算符在后)	... --
	逻辑非	! ...
	按位非	~ ...
	一元加法	+ ...
	一元减法	- ...
	前置递增	++ ...
	前置递减	-- ...
	typeof	typeof ...
	void	void ...
	delete	delete ...
	await	await ...
	幂	... ** ...

乘法	... * ...
除法	... / ...
取模	... % ...
加法	... + ...
减法	... - ...
按位左移	... << ...
按位右移	... >> ...
无符号右移	... >>> ...
小于	... < ...
小于等于	... <= ...
大于	... > ...
大于等于	... >= ...
in	... in ...
instanceof	... instanceof ...
等号	... == ...
非等号	... != ...
全等号	... === ...
非全等号	... !== ...
按位与	... & ...
按位异或	... ^ ...
按位或	... | ...

逻辑与	... && ...
逻辑或	... || ...
条件运算符	... ? ... : ...
赋值	... = ...
	... += ...
	... -= ...
	... *= ...
	... /= ...
	... %= ...
	... <<= ...
	... >>= ...
	... >>>= ...
	... &= ...
	... ^= ...
	... |= ...
yield	yield ...
yield*	yield* ...
展开运算符	... ...
逗号	... , ...

谢谢观看！

如果你有自己难以解决的问题，请发送邮件到 xingxingclassroom@aliyun.com，并备注

星星课堂问题咨询这几个字+问题介绍为邮件标题，谢谢。

我是星星课堂老师：周小周