运算符优先级

运算符的优先级决定了表达式中运算执行的先后顺序，优先级高的运算符最先被执行。

下面是一个简单的例子：

```
3 + 4 * 5 // 计算结果为 23
```

乘法运算符 ("*")比起加法运算符("+")有着更高的优先级，所以它会被最先执行。

结合性

结合性决定了拥有相同优先级的运算符的执行顺序。考虑下面这个表达式：

```
a OP b OP c
```

左结合(从左到右计算)相当于把左边的子表达式加上小括号`(a OP b) OP c`，类似的，右关联(从右到左计算)相当于`a OP (b OP c)`。赋值运算符是右关联的，所以你可以这么写：

```
a = b = 5;
```

结果 `a` 和 `b` 的值都会成为5。这是因为赋值运算符的返回结果就是赋值运算符右边的那个值，具体过程是：`b`被赋值为5，然后`a`也被赋值为 `b=5` 的返回值，也就是5。

汇总表

下面的表将所有运算符按照优先级的不同从高到低排列。

优先级	运算类型	关联性	运算符
20	圆括号	n/a	`( … )`
19	成员访问	从左到右	`… . …`
	需计算的成员访问	从左到右	`… [ … ]`
	`new` (带参数列表)	n/a	`new … ( … )`
	函数调用	从左到右	`… ( … )`
18	new (无参数列表)	从右到左	`new …`
17	后置递增(运算符在后)	n/a	`… ++`
	后置递减(运算符在后)		`… --`
16	逻辑非	从右到左	`! …`
	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 …`
15	幂	从右到左	`… ** …`
14	乘法	从左到右	`… * …`
	除法		`… / …`

	取模		… % …
13	加法	从左到右	… + …
	减法		… - …
12	按位左移	从左到右	… << …
	按位右移		… >> …
	无符号右移		… >>> …
11	小于	从左到右	… < …
	小于等于		… <= …
	大于		… > …
	大于等于		… >= …
	in		… in …
	instanceof		… instanceof …
10	等号	从左到右	… == …
	非等号		… != …
	全等号		… === …
	非全等号		… !== …
9	按位与	从左到右	… & …
8	按位异或	从左到右	… ^ …
7	按位或	从左到右	… | …
6	逻辑与	从左到右	… && …
5	逻辑或	从左到右	… || …
4	条件运算符	从右到左	… ? … : …
3	赋值	从右到左	… = …
			… += …
			… -= …
			… *= …
			… /= …
			… %= …
			… <<= …

			… >>= …
			… >>>= …
			… &= …
			… ^= …
			… |= …
2	yield	从右到左	yield …
	yield*		yield* …
1	展开运算符	n/a	... …
0	逗号	从左到右	… , …