

JS 运算符

从入门到工作：JS 全解

版权声明

本内容版权属杭州饥人谷教育科技有限公司（简称饥人谷）所有。

任何媒体、网站或个人未经本网协议授权不得转载、链接、转贴，
或以其他方式复制、发布和发表。

已获得饥人谷授权的媒体、网站或个人在使用时须注明「资料来源：饥人谷」。

对于违反者，饥人谷将依法追究责任。

联系方式

如果你想要购买本课程
请微信联系 **xiedaimala02** 或 **xiedaimala03**

如果你发现有人盗用本课程
请微信联系 **xiedaimala02** 或 **xiedaimala03**

目录

- 算术运算符
- 比较运算符
- 布尔运算符
- 二进制位运算符
- 其他运算符

算术运算符

数学运算

算术运算符

- **number** 运算

- ✓ 加减乘除
- ✓ 余数 `x % 7`
- ✓ 指数 `x ** 3`
- ✓ 自增自减 `x++ / ++x / x-- / --x`
- ✓ 求值运算符 `+x`
- ✓ 负数运算符 `-x`

- **string** 运算

- ✓ 连接运算 `'123' + '456'`

尽量少用自增自减

因为容易你和别人都记错

不同类型不要加起来

把 1 和 '2' 加起来是几个意思

比较运算符

✓ >

✓ <

✓ >=

✓ <=

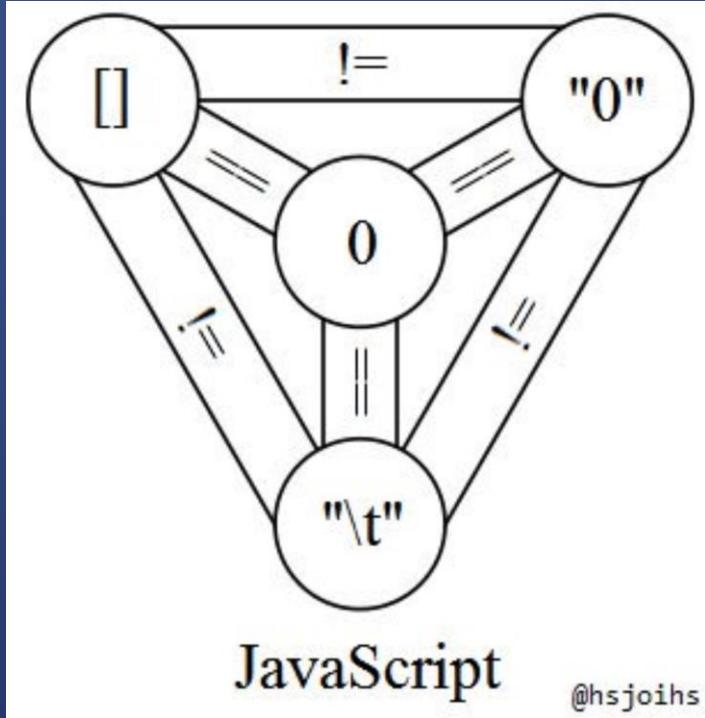
✓ ==

✓ !=

✓ ===

✓ !==

JS 三位一体



`0 == []`

`0 == '0'`

`0 == '\t'`

但是右边三个互不相等

忠告：永远不要使用 `==`，用 `===` 代替
`==` 的问题在于，它总是自作聪明（自动类型转换）

$x == y$ 真值表

	false	0	""	[]	0	"0"	[0]	[1]	""	-1	1	true	-1	null	undefined	NaN	{} ())	"true"	"false"	-Infinity	Infinity	NaN
false	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
""	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
"0"	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[0]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
""	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
true	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
null	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
undefined	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Infinity	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-Infinity	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
"false"	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
"true"	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
{} ())	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NaN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

令人难以理解

[]==false但不是 falsy

[]==false但{}却不是

[[]]==false

$x == y$ 真值表

true	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
false	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
1	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
0	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
-1	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
"true"	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
"false"	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
"1"	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
"0"	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
"-1"	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
""	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
null	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
undefined	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
Infinity	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
-Infinity	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
[]	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
{}	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
[[]]	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
[0]	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
[1]	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
NaN	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN

没有任何费解

- 基本类型看值是否相等
- 对象看地址是否相等

[] != []

{ } != { }

NaN != NaN

唯一特例，强行记忆一下

布尔运算符

- 或且非

- ✓ `||`
- ✓ `&&`
- ✓ `!`

- 短路逻辑

- ✓ `console && console.log && console.log('hi')`
- ✓ 以防 `console` 不存在报错
- ✓ `a = a || 100`
- ✓ `a` 的保底值

二进制运算符

- 或、与、否

- ✓ | 两个位都为0，则结果为0，否则为1

- ✓ &

- ✓ ~

- 异或

- ✓ ^

- ✓ 两个位相同，则结果为0，否则为1

- 左移右移

- ✓ << 和 >>

- 头部补零的右移运算符

- ✓ >>>

平时工作很少用
面试喜欢问

参考

- 位运算符在JS中的妙用

使用与运算符判断奇偶

- 代码

- ✓ 偶数 & 1 = 0
- ✓ 奇数 & 1 = 1

使用~, >>, <<, >>>, | 来取整

- 代码

- ✓ `console.log(~~ 6.83) // 6`
- ✓ `console.log(6.83 >> 0) // 6`
- ✓ `console.log(6.83 << 0) // 6`
- ✓ `console.log(6.83 | 0) // 6`
- ✓ `console.log(6.83 >>> 0) // 6`

使用^来交换 a b 的值

- 代码

```
var a = 5
var b = 8
a ^= b
b ^= a
a ^= b
console.log(a)    // 8
console.log(b)    // 5
```

位运算用得很少容易忘

面试之前重新看看即可

奇葩其他运算符符

一个比一个奇葩

点运算符

- **语法**

- ✓ 对象.属性名 = 属性值

- **作用**

- ✓ 读取对象的属性值

- **有个疑问**

- ✓ 不是对象，为什么也可以有属性? 'a-b-c'.split('-')
- ✓ JS 有特殊逻辑，点前面不是对象，就把它封装成对象
- ✓ number 会变成 Number 对象
- ✓ string 会变成 String 对象
- ✓ bool 会变成 Boolean 对象
- ✓ 程序员从来不用这三种对象，只用简单类型

void 运算符

- 语法

- ✓ void 表达式或语句

- 作用

- ✓ 求表达式的值，或执行语句
- ✓ 然后 void 的值总是为 undefined

- 需求

- ✓ 点击
- ✓ return 假值可以阻止默认动作
- ✓ 文字
- ✓ 改用 void 可以炫技

逗号运算符

- **语法**

- ✓ 表达式1, 表达式2, ..., 表达式n

- **作用**

- ✓ 将表达式 n 的值作为整体的值

- **使用**

- ✓ `let a = (1,2,3,4,5)`
 - ✓ 那么 a 的值就是 5，奇葩吧？
 - ✓ `let f = (x) => (console.log('平方值为'), x*x)`
 - ✓ 注意上面的括号不能省

运算符优先级

先算什么，后算什么

优先级是什么

- 不同运算符

- ✓ $1 + 2 * 3$ 是 $(1 + 2) * 3$ 还是 $1 + (2 * 3)$
- ✓ $! a === 1$ 是 $(! a) === 1$ 还是 $! (a === 1)$
- ✓ `new Person().sayHi()` 是什么意思

- 相同运算符

- ✓ 从左到右 $a + b + c$
- ✓ 从右到左 $a = b = c = d$

- 优先级就是先算什么后算什么

- ✓ 具体规则想知道吗？
- ✓ 你：想。
- ✓ 不，你不想！看看[这里](#)就知道为什么。

优先级汇总

• 汇总表位于 MDN

- ✓ 一共有20个运算符
- ✓ 怎么记忆呢

• 技巧

- ✓ 圆括号优先级最高
- ✓ 会用圆括号就行
- ✓ 其他一律不记

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
	带计算的成员访问	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	, ...

面试技巧

- 本节课有两个推荐大家不学
 - ✓ == 不学
 - ✓ 优先级不学
- 面试遇到怎么办
 - ✓ 想面试官说明为什么不学，以及你的态度
 - ✓ 如果面试官非要你答，直接跳过
 - ✓ 跳过不影响面试表现吗？你可以在其他题表现
 - ✓ 没必要在这种烂题上表现

再见

JS 基本语法基本学会，下节课总结一下