TypeScript 变量声明 TypeScript 条件语句

TypeScript 运算符

运算符用于执行程序代码运算，会针对一个以上操作数项目来进行运算。

考虑以下计算：

```
7 + 5 = 12
```

以上实例中 7、5 和 12 是操作数。

运算符 `+` 用于加值。

运算符 `=` 用于赋值。

TypeScript 主要包含以下几种运算：

- 算术运算符
- 逻辑运算符
- 关系运算符
- 按位运算符
- 赋值运算符
- 三元/条件运算符
- 字符串运算符
- 类型运算符

算术运算符

假定 **y=5**，下面的表格解释了这些算术运算符的操作：

运算符	描述	例子	x 运算结果	y 运算结果
+	加法	x=y+2	7	5
-	减法	x=y-2	3	5
*	乘法	x=y*2	10	5
/	除法	x=y/2	2.5	5
%	取模（余数）	x=y%2	1	5
++	自增	x=++y	6	6
		x=y++	5	6

--	自减	x=--y	4	4
		x=y--	5	4

实例

```
var num1:number = 10
var num2:number = 2
var res:number = 0
res = num1 + num2
console.log("加: "+res);
res = num1 - num2;
console.log("减: "+res)
res = num1*num2
console.log("乘: "+res)
res = num1/num2
console.log("除: "+res)
res = num1%num2
console.log("余数: "+res)
num1++
console.log("num1 自增运算: "+num1)
num2--
console.log("num2 自减运算: "+num2)
```

使用 `tsc` 命令编译以上代码得到如下 JavaScript 代码：

```
var num1 = 10;
var num2 = 2;
var res = 0;
res = num1 + num2;
console.log("加: " + res);
res = num1 - num2;
console.log("减: " + res);
res = num1 * num2;
console.log("乘: " + res);
res = num1 / num2;
console.log("除: " + res);
res = num1 % num2;
console.log("余数: " + res);
num1++;
console.log("num1 自增运算: " + num1);
num2--;
console.log("num2 自减运算: " + num2);
```

执行以上 JavaScript 代码，输出结果为：

```
加:        12
减: 8
乘:    20
除:   5
余数:   0
```

```
num1 自增运算: 11
num2 自减运算: 1
```

关系运算符

关系运算符用于计算结果是否为 true 或者 false。

x=5，下面的表格解释了关系运算符的操作：

运算符	描述	比较	返回值
==	等于	x==8	*false*
		x==5	*true*
!=	不等于	x!=8	*true*
>	大于	x>8	*false*
<	小于	x<8	*true*
>=	大于或等于	x>=8	*false*
<=	小于或等于	x<=8	*true*

实例

```
var num1:number = 5;
var num2:number = 9;
console.log("num1 的值为: "+num1);
console.log("num2 的值为:"+num2);
var res = num1>num2
console.log("num1 大于n num2: "+res)
res = num1<num2
console.log("num1 小于 num2: "+res)
res = num1>=num2
console.log("num1 大于或等于 num2: "+res)
res = num1<=num2
console.log("num1 小于或等于 num2: "+res)
res = num1==num2
console.log("num1 等于 num2: "+res)
res = num1!=num2
console.log("num1 不等于 num2: "+res)
```

使用 `tsc` 命令编译以上代码得到如下 JavaScript 代码：

```
var num1 = 5;
var num2 = 9;
console.log("num1 的值为: " + num1);
console.log("num2 的值为:" + num2);
var res = num1 > num2;
```

```
console.log("num1 大于n num2: " + res);
res = num1 < num2;
console.log("num1 小于 num2: " + res);
res = num1 >= num2;
console.log("num1 大于或等于 num2: " + res);
res = num1 <= num2;
console.log("num1 小于或等于 num2: " + res);
res = num1 == num2;
console.log("num1 等于 num2: " + res);
res = num1 != num2;
console.log("num1 不等于 num2: " + res);
```

执行以上 JavaScript 代码，输出结果为：

```
num1 的值为: 5
num2 的值为:9
num1 大于n num2: false
num1 小于 num2: true
num1 大于或等于  num2: false
num1 小于或等于 num2: true
num1 等于 num2: false
num1 不等于 num2: true
```

逻辑运算符

逻辑运算符用于测定变量或值之间的逻辑。

给定 x=6 以及 y=3，下表解释了逻辑运算符：

运算符	描述	例子
&&	and	(x < 10 && y > 1) 为 true
||	or	(x==5 || y==5) 为 false
!	not	!(x==y) 为 true

实例

```
var avg:number = 20;
var percentage:number = 90;
console.log("avg 值为: "+avg+" ,percentage 值为: "+percentage);
var res:boolean = ((avg>50)&&(percentage>80));
console.log("(avg>50)&&(percentage>80): ",res);
var res:boolean = ((avg>50)||(percentage>80));
console.log("(avg>50)||(percentage>80): ",res);
var res:boolean=!((avg>50)&&(percentage>80));
console.log("!((avg>50)&&(percentage>80)): ",res);
```

使用 `tsc` 命令编译以上代码得到如下 JavaScript 代码：

```
var avg = 20;
var percentage = 90;
console.log("avg 值为: " + avg + " ,percentage 值为: " + percentage);
var res = ((avg > 50) && (percentage > 80));
console.log("(avg>50)&&(percentage>80): ", res);
var res = ((avg > 50) || (percentage > 80));
console.log("(avg>50)||(percentage>80): ", res);
var res = !((avg > 50) && (percentage > 80));
console.log("!((avg>50)&&(percentage>80)): ", res);
```

执行以上 JavaScript 代码，输出结果为：

```
avg 值为: 20 ,percentage 值为: 90
(avg>50)&&(percentage>80):  false
(avg>50)||(percentage>80):  true
!((avg>50)&&(percentage>80)):  true
```

短路运算符(&& 与 ||)

&& 与 || 运算符可用于组合表达式。 && 运算符只有在左右两个表达式都为 true 时才返回 true。

考虑以下实例：

```
var a = 10
var result = ( a<10 && a>5)
```

以上实例中 a < 10 与 a > 5 是使用了 && 运算符的组合表达式，第一个表达式返回了 false，由于 && 运算需要两个表达式都为 true，所以如果第一个为 false，就不再执行后面的判断(a > 5 跳过计算)，直接返回 false。

|| 运算符只要其中一个表达式为 true ，则该组合表达式就会返回 true。

考虑以下实例：

```
var a = 10
var result = ( a>5 || a<10)
```

以上实例中 a > 5 与 a < 10 是使用了 || 运算符的组合表达式，第一个表达式返回了 true，由于 || 组合运算只需要一个表达式为 true，所以如果第一个为 true，就不再执行后面的判断(a < 10 跳过计算)，直接返回 true。

位运算符

位操作是程序设计中对位模式按位或二进制数的一元和二元操作。

; 0001

运算符	描述	例子	类似于	结果	十进制	
&	AND，按位与处理两个长度相同的二进制数，两个相应的	x = 5 & 1	0101 & 0001	0001	0001	1

	二进位都为 1，该位的结果值才为 1，否则为 0。					
|	OR，按位或处理两个长度相同的二进制数，两个相应的二进位中只要有一个为 1，该位的结果值为 1。	x = 5 | 1	0101 | 0001	0101	5	
~	取反，取反是一元运算符，对一个二进制数的每一位执行逻辑反操作。使数字 1 成为 0，0 成为 1。	x = ~ 5	~0101	1010	-6	
^	异或，按位异或运算，对等长二进制模式按位或二进制数的每一位执行逻辑异按位或操作。操作的结果是如果某位不同则该位为 1，否则该位为 0。	x = 5 ^ 1	0101 ^ 0001	0100	4	
<<	左移，把 << 左边的运算数的各二进位全部左移若干位，由 << 右边的数指定移动的位数，高位丢弃，低位补 0。	x = 5 << 1	0101 << 1	1010	10	
>>	右移，把 >> 左边的运算数的各二进位全部右移若干位，>> 右边的数指定移动的位数。	x = 5 >> 1	0101 >> 1	0010	2	
>>>	无符号右移，与有符号右移位类似，除了左边一律使用0 补位。	x = 2 >>> 1	0010 >>> 1	0001	1	

实例

```
var a:number = 2; // 二进制 10
var b:number = 3; // 二进制 11
var result;
result = (a & b);
console.log("(a & b) => ",result)
result = (a | b);
console.log("(a | b) => ",result)
```

```
result = (a ^ b);
console.log("(a ^ b) => ",result);
result = (~b);
console.log("(~b) => ",result);
result = (a << b);
console.log("(a << b) => ",result);
result = (a >> b);
console.log("(a >> b) => ",result);
result = (a >>> 1);
console.log("(a >>> 1) => ",result);
```

使用 `tsc` 命令编译以上代码得到如下 JavaScript 代码：

```
var a = 2; // 二进制 10
var b = 3; // 二进制 11
var result;
result = (a & b);
console.log("(a & b) => ", result);
result = (a | b);
console.log("(a | b) => ", result);
result = (a ^ b);
console.log("(a ^ b) => ", result);
result = (~b);
console.log("(~b) => ", result);
result = (a << b);
console.log("(a << b) => ", result);
result = (a >> b);
console.log("(a >> b) => ", result);
result = (a >>> 1);
console.log("(a >>> 1) => ", result);
```

执行以上 JavaScript 代码，输出结果为：

```
(a & b) =>  2
(a | b) =>  3
(a ^ b) =>  1
(~b) =>  -4
(a << b) =>  16
(a >> b) =>  0
(a >>> 1) =>  1
```

赋值运算符

赋值运算符用于给变量赋值。

给定 **x=10** 和 **y=5**，下面的表格解释了赋值运算符：

运算符	例子	实例	x 值
= (赋值)	x = y	x = y	x = 5
+= (先进行加运算后赋值)	x += y	x = x + y	x = 15

-= (先进行减运算后赋值)	x -= y	x = x - y	x = 5
*= (先进行乘运算后赋值)	x *= y	x = x * y	x = 50
/= (先进行除运算后赋值)	x /= y	x = x / y	x = 2

类似的逻辑运算符也可以与赋值运算符联合使用：<<=, >>=, >>=, &=, |= 与 ^=。

实例

```
var a: number = 12
var b:number = 10
a = b
console.log("a = b: "+a)
a += b
console.log("a+=b: "+a)
a -= b
console.log("a-=b: "+a)
a *= b
console.log("a*=b: "+a)
a /= b
console.log("a/=b: "+a)
a %= b
console.log("a%=b: "+a)
```

使用 `tsc` 命令编译以上代码得到如下 JavaScript 代码：

```
var a = 12;
var b = 10;
a = b;
console.log("a = b: " + a);
a += b;
console.log("a+=b: " + a);
a -= b;
console.log("a-=b: " + a);
a *= b;
console.log("a*=b: " + a);
a /= b;
console.log("a/=b: " + a);
a %= b;
console.log("a%=b: " + a);
```

执行以上 JavaScript 代码，输出结果为：

```
a = b: 10
a+=b: 20
a-=b: 10
a*=b: 100
a/=b: 10
a%=b: 0
```

三元运算符 (?)

三元运算有 3 个操作数，并且需要判断布尔表达式的值。该运算符的主要是决定哪个值应该赋值给变量。

```
Test ? expr1 : expr2
```

- Test – 指定的条件语句
- expr1 – 如果条件语句 Test 返回 true 则返回该值
- expr2 – 如果条件语句 Test 返回 false 则返回该值

让我们看下以下实例：

```
var num:number = -2
var result = num > 0 ? "大于 0" : "小于 0，或等于 0"
console.log(result)
```

实例中用于判断变量是否大于 0。

使用 tsc 命令编译以上代码得到如下 JavaScript 代码：

```
var num = -2;
var result = num > 0 ? "大于 0" : "小于 0，或等于 0";
console.log(result);
```

以上实例输出结果如下：

```
小于 0，或等于 0
```

类型运算符

typeof 运算符

typeof 是一元运算符，返回操作数的数据类型。

查看以下实例:

```
var num = 12
console.log(typeof num); //输出结果: number
```

使用 tsc 命令编译以上代码得到如下 JavaScript 代码：

```
var num = 12;
console.log(typeof num); //输出结果: number
```

以上实例输出结果如下：

```
number
```

instanceof

instanceof 运算符用于判断对象是否为指定的类型，后面章节我们会具体介绍它。

其他运算符

负号运算符(-)

更改操作数的符号，查看以下实例：

```
var x:number = 4
var y = -x;
console.log("x 值为: ",x); // 输出结果 4
console.log("y 值为: ",y); // 输出结果 -4
```

使用 tsc 命令编译以上代码得到如下 JavaScript 代码：

```
var x = 4;
var y = -x;
console.log("x 值为: ", x); // 输出结果 4
console.log("y 值为: ", y); // 输出结果 -4
```

以上实例输出结果如下：

```
x 值为:  4
y 值为:  -4
```

字符串运算符: 连接运算符 (+)

+ 运算符可以拼接两个字符串，查看以下实例：

```
var msg:string = "RUNOOB"+".COM"
console.log(msg)
```

使用 tsc 命令编译以上代码得到如下 JavaScript 代码：

```
var msg = "RUNOOB" + ".COM";
console.log(msg);
```

以上实例输出结果如下：

```
RUNOOB.COM
```

← TypeScript 变量声明 TypeScript 条件语句 →

点我分享笔记