

## JavaScript 常用运算符和操作符总结

分类	操作符
算术操作	+、-、*、/、% (取余)
一元操作	++、--
逻辑运算	!、&&、
关系运算	<、>、<=、>=、!=、!==、==、===
赋值操作	=、复合赋值 (+ =、- =、* =、/ =)

### 一、算术运算符 (案例中 y=5)

运算符	描述	例子	结果
+	加	x=y+2	7
-	减	x=y-2	3
*	乘	x=y*2	10
/	除	x=y/2	2.5
%	取余	x=y%2	1

#### 注意：

- (1) “+” 运算符还能用于把文本值或字符串连接起来，如果要把两个或以上的字符串连接起来，可以用 “+” 运算符；抑或想把别的数据类型转化为字符串，也可以用 “+” 运算符把该数据类型跟字符串相加

例如：

```
<!DOCTYPE html>
```

```
<html>
```

```
<head lang="en">

  <meta charset="UTF-8">

  <title>运算符</title>

</head>

<body>

  <script type="text/javascript">

    var bool=true,string1="hello",string2="world",num=123;

    var

result1=bool+string1,result2=num+string1,result3=string1+string2;

    console.log("布尔值和字符串相加结果是："+result1+" ,相加后的数
据类型是："+typeof result1);

    console.log("数字和字符串相加结果是："+result2+" ,相加后的数据
类型是："+typeof result2);

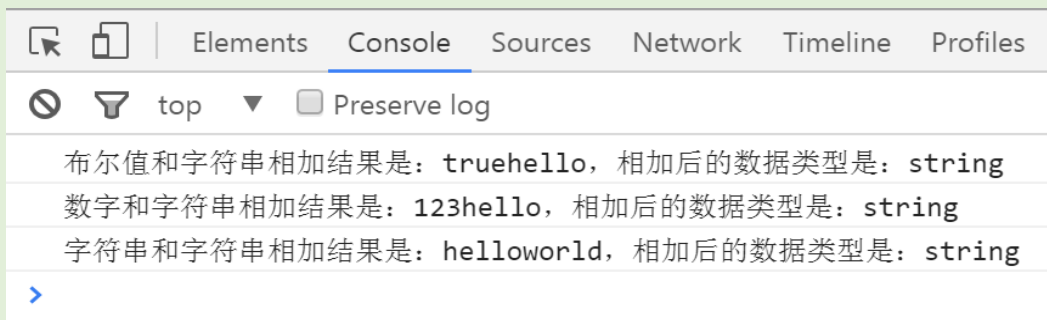
    console.log("字符串和字符串相加结果是："+result3+" ,相加后的数
据类型是："+typeof result3);

  </script>

</body>

</html>
```

输出结果如下：



(2) 取余，由百分号 (%) 表示

例如：

```
<!DOCTYPE html>

<html>

  <head lang="en">

    <meta charset="UTF-8">

    <title>运算符</title>

  </head>

  <body>

    <script type="text/javascript">

      var x=10,y=3,z=0;

      var result1=x%y,result2=x%x,result3=x%z;

      console.log("10%3="+result1);

      console.log("10%10="+result2);

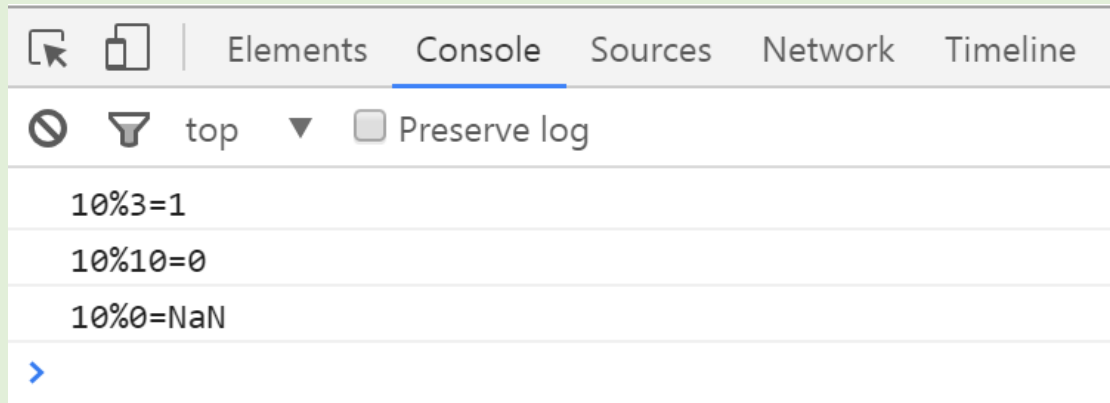
      console.log("10%0="+result3);

    </script>

  </body>

</html>
```

输出结果如下：



## 二、一元操作符

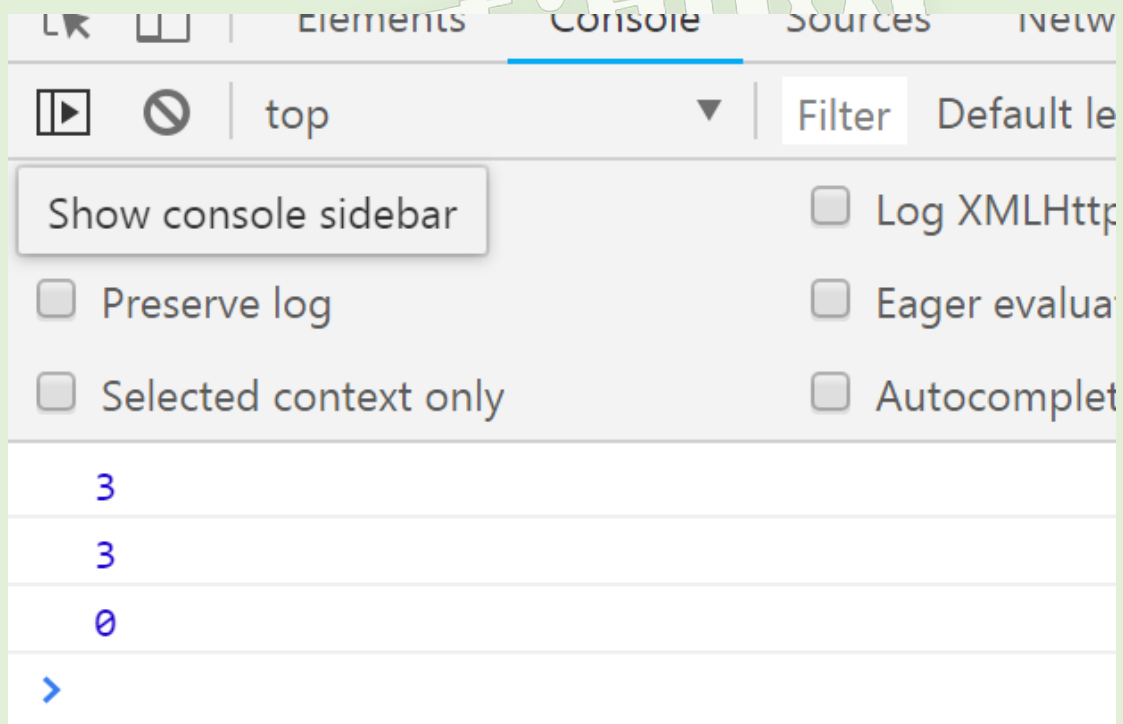
(1) ++/--变量，赋值(+1或-1)后再运算

例如：

```
<!DOCTYPE html>
<html>
  <head lang="en">
    <meta charset="UTF-8">
    <title>运算符</title>
  </head>
  <body>
    <script type="text/javascript">
      var x=1,y=2;
      ++x;--y;
      var result=(++x)+(--y);
      console.log(result);
      console.log(x);
```

```
        console.log(y);  
    </script>  
</body>  
</html>
```

输出结果是：



分析：

++/--在变量的前面，变量先自身+1 或-1，然后再进行运算；x 进行两次自增加值是 3，y 进行两次自减值是 0，故相加后结果是 3

( 2 ) 变量++/--，运算后再赋值 ( +1 或-1 )

例如：

```
<!DOCTYPE html>  
  
<html>  
  
    <head lang="en">
```

```
<meta charset="UTF-8">

<title>运算符</title>

</head>

<body>

  <script type="text/javascript">

    var x=1,y=2;

    var result=(++x)+(y--);

    console.log(result);

    console.log(x);

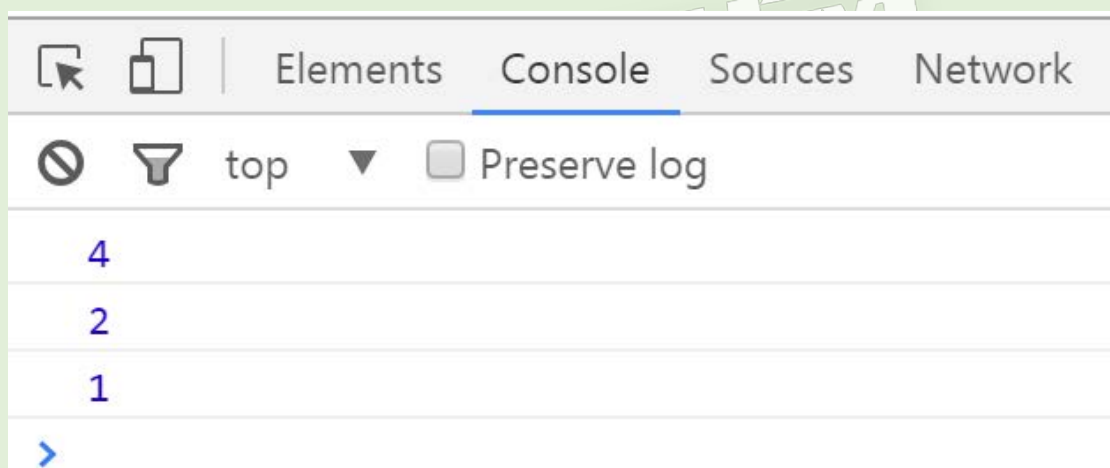
    console.log(y);

  </script>

</body>

</html>
```

输出结果是：



分析：

++/--在变量的后面，变量先进行运算，然后再赋值；(++x)+(y--)的

运算中，++x 先赋值再运算，x 就是 2；y--先参与运算再赋值，故 y 是 2，所以(++x)+(y--)的值是 4，x 是 2，y 是 1

### 三、复合赋值运算

(1) 复合赋值运算是由算术操作符实现的，是算术操作的缩写形式，如：

```
var a=10;
```

```
a=a+10;
```

第二行代码由复合赋值运算来写的话就是：

```
a+=10;
```

(2) 主要的算术运算的复合赋值运算符如下：

加法：+=

减法：-=

乘法：\*=

除法：/=

取余：%=

### 四、逻辑运算符

X	Y	X&&Y	X  Y	!X
false	false	false	false	true
false	true	false	true	true
true	false	false	true	false
true	true	true	true	false

#### 关于“短路逻辑”的问题

(1) 在&&运算符中

- 
- 如果第一个运算公式为真，那么再去看后面的运算公式是否为真；
  - 如果第一个运算公示为假，那么后面的运算公示就没必要去看，这个逻辑肯定为假

例如：

```
<!DOCTYPE html>

<html>

  <head lang="en">

    <meta charset="UTF-8">

    <title>逻辑运算符</title>

  </head>

  <body>

    <script type="text/javascript">

      var x=1,y=2;

      var result=++x<0&&y++>0;

      console.log(result);

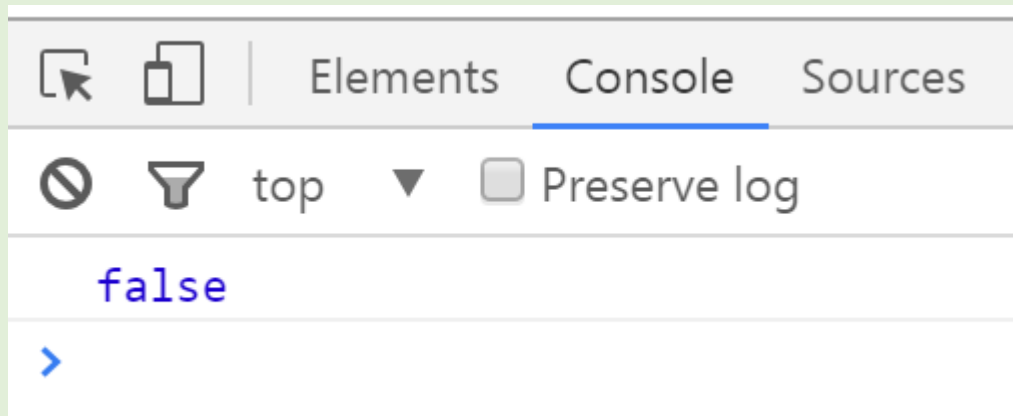
    </script>

  </body>

</html>
```

输出结果是：





(2) 在||运算中

- 如果第一个运算公式为真，那么后面的运算公式没必要去看，这个逻辑运算肯定为真
- 如果第一个运算公式为假，那么再去看后面的运算公式是否为真

例如：

```
<!DOCTYPE html>

<html>

  <head lang="en">

    <meta charset="UTF-8">

    <title>逻辑运算符</title>

  </head>

  <body>

    <script type="text/javascript">

      var x=1,y=2;

      var result=++x<0||y++>0;

      console.log(result);

    </script>
```

```
</body>

</html>
```

输出结果是：

