### Jscript

#### 按位“与”运算符 (&)

对两个表达式执行按位“与”。

result **=** expression1 **&** expression2

#### 参数

result

任何[变量](http://javascript.jz123.cn/html/glossary.htm#defvariablejs)。

expression1

任何[表达式](http://javascript.jz123.cn/html/glossary.htm#defexpressionjs)。

expression2

任何表达式。

#### 说明

**&** 运算符查看两个表达式的二进制表示法的值，并执行按位“与”操作。该操作的结果如下所示：

0101   (expression1)

1100   (expression2)

----

0100   (result)

任何时候，只要两个表达式的某位都为 1，则结果的该位为 1。否则，结果的该位为 0。

### Java

#### 说说&和&&的区别?

&和&&都可以用作逻辑与的运算符，表示逻辑与（and），当运算符两边的表达式的结果都为true时，整个运算结果才为true，否则，只要有一方为false，则结果为false。

&&还具有短路的功能，即如果第一个表达式为false，则不再计算第二个表达式，例如，对于if(str != null && !str.equals(“”))表达式，当str为null时，后面的表达式不会执行，所以不会出现NullPointerException如果将&&改为&，则会抛出NullPointerException异常。

If(x==33 & ++y>0) y会增长，If(x==33 && ++y>0)不会增长

&还可以用作位运算符，当&操作符两边的表达式不是boolean类型时，&表示按位与操作，我们通常使用0x0f来与一个整数进行&运算，来获取该整数的最低4个bit位，例如，0x31 & 0x0f的结果为0x01。

#### 当&操作符两边的表达式不是boolean类型时，&表示按位与操作

通常使用0x0f (0x表示该数值用十六进制表示) 来与一个整数进行&运算，来获取该整数的最低4个bit位，例如，0x31 & 0x0f的结果为0x01

。

String str="0F"; //str是16进制数的字符串

**int** i=Integer.*parseInt*(str,16); //16进制转化为10进制

String str2=Integer.*toBinaryString*(i); //10进制转化为2进制

16进制 "0F" 转换为 2进制 后为 "1111" .

所以"0F"也就是"1111"与一个整数进行 &运算 ，获得的是该整数的最低4个bit 位 (原理参见Jscript中的[说明])

顺便提下：

Integer.*toHexString*(123); //10进制转化为16进制