

元字符	描述
\	将下一个字符标记为一个特殊字符、或一个原义字符、或一个向后引用、或一个八进制转义符。例如，“\\n”匹配\n。“\n”匹配换行符。序列“\\”匹配“\”而“\(”则匹配“（”。即相当于多种编程语言中都有的“转义字符”的概念。
^	匹配输入字符串的开始位置。如果设置了 RegExp 对象的 Multiline 属性，^也匹配“\n”或“\r”之后的位置。
\$	匹配输入字符串的结束位置。如果设置了 RegExp 对象的 Multiline 属性，\$也匹配“\n”或“\r”之前的位置。
*	匹配前面的子表达式任意次。例如，zo*能匹配“z”，“zo”以及“zoo”。*等价于{0,}。
+	匹配前面的子表达式一次或多次(大于等于 1 次)。例如，“zo+”能匹配“zo”以及“zoo”，但不能匹配“z”。+等价于{1,}。
?	匹配前面的子表达式零次或一次。例如，“do(es)?”可以匹配“do”或“does”中的“do”。?等价于{0,1}。
{n}	n 是一个非负整数。匹配确定的 n 次。例如，“o{2}”不能匹配“Bob”中的“o”，但是能匹配“food”中的两个 o。
{n,}	n 是一个非负整数。至少匹配 n 次。例如，“o{2,}”不能匹配“Bob”中的“o”，但能匹配“foooooo”中的所有 o。“o{1,}”等价于“o+”。“o{0,}”则等价于“o*”。
{n,m}	m 和 n 均为非负整数，其中 n<=m。最少匹配 n 次且最多匹配 m 次。例如，“o{1,3}”将匹配“foooooo”中的前三个 o。“o{0,1}”等价于“o?”。请注意在逗号和两个数之间不能有空格。
?	当该字符紧跟在任何一个其他限制符 (*, +, ?, {n}, {n,}, {n,m}) 后面时，匹配模式是非贪婪的。非贪婪模式尽可能少的匹配所搜索的字符串，而默认的贪婪模式则尽可能多的匹配所搜索的字符串。例如，对于字符串“oooo”，“o+?”将匹配单个“o”，而“o+”将匹配所有“o”。
. 点	匹配除“\r\n”之外的任何单个字符。要匹配包括“\r\n”在内的任何字符，请使用像“[\s\S]”的模式。
(pattern)	匹配 pattern 并获取这一匹配。所获取的匹配可以从产生的 Matches 集合得到，在 VBScript 中使用 SubMatches 集合，在 JScript 中则使用 \$0...\$9 属性。要匹配圆括号字符，请使用“\(”或“\)”。
(?:pattern)	匹配 pattern 但不获取匹配结果，也就是说这是一个非获取匹配，不进行存储供以后使用。这在使用或字符“()”来组合一个模式的各个部分是很有用。例如“industr(?:y ies)”就是一个比“industry industries”更简略的表达式。
(?=pattern)	正向肯定预查，在任何匹配 pattern 的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配不需要获取供以后使用。例如，“Windows(=95 98 NT 2000)”能匹配“Windows2000”

	<p>中的“Windows”，但不能匹配“Windows3.1”中的“Windows”。预查不消耗字符，也就是说，在一个匹配发生后，在最后一次匹配之后立即开始下一次匹配的搜索，而不是从包含预查的字符之后开始。</p> <p>正向否定预查，在任何不匹配 pattern 的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配不需要获取供以后使用。例如“Windows(?!95 98 NT 2000)”能匹配“Windows3.1”中的“Windows”，但不能匹配“Windows2000”中的“Windows”。</p>
(?!pattern)	
(?<=pattern)	<p>反向肯定预查，与正向肯定预查类似，只是方向相反。例如，“(?<=95 98 NT 2000)Windows”能匹配“2000Windows”中的“Windows”，但不能匹配“3.1Windows”中的“Windows”。</p>
(?<!pattern)	<p>反向否定预查，与正向否定预查类似，只是方向相反。例如“(?<!95 98 NT 2000)Windows”能匹配“3.1Windows”中的“Windows”，但不能匹配“2000Windows”中的“Windows”。</p>
x y	<p>匹配 x 或 y。例如，“z food”能匹配“z”或“food”或“zood”(此处请谨慎)。“(z f)ood”则匹配“zood”或“food”。</p>
[xyz]	<p>字符集合。匹配所包含的任意一个字符。例如，“[abc]”可以匹配“plain”中的“a”。</p>
[^xyz]	<p>负值字符集合。匹配未包含的任意字符。例如，“[^abc]”可以匹配“plain”中的“plin”。</p> <p>字符范围。匹配指定范围内的任意字符。例如，“[a-z]”可以匹配“a”到“z”范围内的任意小写字母字符。</p>
[a-z]	<p>注意:只有连字符在字符组内部时,并且出现在两个字符之间时,才能表示字符的范围; 如果出字符组的开头,则只能表示连字符本身.</p>
[^a-z]	<p>负值字符范围。匹配任何不在指定范围内的任意字符。例如，“[^a-z]”可以匹配任何不在“a”到“z”范围内的任意字符。</p>
\b	<p>匹配一个单词边界，也就是指单词和空格间的位置（即正则表达式的“匹配”有两种概念，一种是匹配字符，一种是匹配位置，这里的\b就是匹配位置的）。例如，“er\b”可以匹配“never”中的“er”，但不能匹配“verb”中的“er”。</p>
\B	<p>匹配非单词边界。“er\B”能匹配“verb”中的“er”，但不能匹配“never”中的“er”。</p>
\cx	<p>匹配由 x 指明的控制字符。例如，\cM 匹配一个 Control-M 或回车符。x 的值必须为 A-Z 或 a-z 之一。否则，将 c 视为一个原义的“c”字符。</p>
\d	<p>匹配一个数字字符。等价于[0-9]。</p>
\D	<p>匹配一个非数字字符。等价于[^0-9]。</p>

<code>\f</code>	匹配一个换页符。等价于 <code>\x0c</code> 和 <code>\cL</code> 。
<code>\n</code>	匹配一个换行符。等价于 <code>\x0a</code> 和 <code>\cJ</code> 。
<code>\r</code>	匹配一个回车符。等价于 <code>\x0d</code> 和 <code>\cM</code> 。
<code>\s</code>	匹配任何不可见字符，包括空格、制表符、换页符等等。等价于 <code>[\f\n\r\t\v]</code> 。
<code>\S</code>	匹配任何可见字符。等价于 <code>[^ \f\n\r\t\v]</code> 。
<code>\t</code>	匹配一个制表符。等价于 <code>\x09</code> 和 <code>\cI</code> 。
<code>\v</code>	匹配一个垂直制表符。等价于 <code>\x0b</code> 和 <code>\cK</code> 。
<code>\w</code>	匹配包括下划线的任何单词字符。类似但不等价于 <code>"[A-Za-z0-9_]"</code> ，这里的“单词”字符使用 Unicode 字符集。
<code>\W</code>	匹配任何非单词字符。等价于 <code>"[^A-Za-z0-9_]"</code> 。
<code>\xn</code>	匹配 <code>n</code> ，其中 <code>n</code> 为十六进制转义值。十六进制转义值必须为确定的两个数字长。例如， <code>"\x41"</code> 匹配 <code>"A"</code> 。 <code>"\x041"</code> 则等价于 <code>"\x04&1"</code> 。正则表达式中可以使用 ASCII 编码。
<code>\num</code>	匹配 <code>num</code> ，其中 <code>num</code> 是一个正整数。对所获取的匹配的引用。例如， <code>"(.)\1"</code> 匹配两个连续的相同字符。
<code>\n</code>	标识一个八进制转义值或一个向后引用。如果 <code>\n</code> 之前至少 <code>n</code> 个获取的子表达式，则 <code>n</code> 为向后引用。否则，如果 <code>n</code> 为八进制数字（0-7），则 <code>n</code> 为一个八进制转义值。
<code>\nm</code>	标识一个八进制转义值或一个向后引用。如果 <code>\nm</code> 之前至少有 <code>nm</code> 个获得子表达式，则 <code>nm</code> 为向后引用。如果 <code>\nm</code> 之前至少有 <code>n</code> 个获取，则 <code>n</code> 为一个后跟文字 <code>m</code> 的向后引用。如果前面的条件都不满足，若 <code>n</code> 和 <code>m</code> 均为八进制数字（0-7），则 <code>\nm</code> 将匹配八进制转义值 <code>nm</code> 。
<code>\nml</code>	如果 <code>n</code> 为八进制数字（0-7），且 <code>m</code> 和 <code>l</code> 均为八进制数字（0-7），则匹配八进制转义值 <code>nml</code> 。
<code>\un</code>	匹配 <code>n</code> ，其中 <code>n</code> 是一个用四个十六进制数字表示的 Unicode 字符。例如， <code>\u00A9</code> 匹配版权符号（©）。
<code>\< \></code>	匹配词（word）的开始（ <code>\<</code> ）和结束（ <code>\></code> ）。例如正则表达式 <code>\<the\></code> 能够匹配字符串 <code>"for the wise"</code> 中的 <code>"the"</code> ，但是不能匹配字符串 <code>"otherwise"</code> 中的 <code>"the"</code> 。注意：这个元字符不是所有的软件都支持的。
<code>\(\)</code>	将 <code>\(</code> 和 <code>\)</code> 之间的表达式定义为“组”（group），并且将匹配这个表达式的字符保存到一个临时区域（一个正则表达式中最多可以保存 9 个），它们可以用 <code>\1</code> 到 <code>\9</code> 的符号来引用。

	将两个匹配条件进行逻辑“或”（Or）运算。例如正则表达式 (him her) 匹配“it belongs to him”和“it belongs to her”，但是不能匹配“it belongs to them.”。注意：这个元字符不是所有的软件都支持的。
+	匹配 1 或多个正好在它之前的那个字符。例如正则表达式 9+ 匹配 9、99、999 等。注意：这个元字符不是所有的软件都支持的。
?	匹配 0 或 1 个正好在它之前的那个字符。注意：这个元字符不是所有的软件都支持的。
{i}	匹配指定数目的字符，这些字符是在它之前的表达式定义的。例如正则表达式 A[0-9]{3} 能够匹配字符“A”后面跟着正好 3 个数字字符的串，例如 A123、A348 等，但是不匹配 A1234。而正则表达式 [0-9]{4,6} 匹配连续的任意 4 个、5 个或者 6 个数字
{i, j}	