

## 表达式全集

字符	描述
<b>\</b>	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， <code>"n"</code> 匹配字符 <code>"n"</code> 。 <code>"\n"</code> 匹配一个换行符。串行 <code>"\\"</code> 匹配 <code>"\"</code> 而 <code>"\"</code> 则匹配 <code>"("</code> 。
<b>^</b>	匹配输入字符串的开始位置。如果设置了RegExp对象的Multiline属性， <code>^</code> 也匹配 <code>"\n"</code> 或 <code>"\r"</code> 之后的位置。
<b>\$</b>	匹配输入字符串的结束位置。如果设置了RegExp对象的Multiline属性， <code>\$</code> 也匹配 <code>"\n"</code> 或 <code>"\r"</code> 之前的位置。
<b>*</b>	匹配前面的子表达式零次或多次。例如， <code>zo*</code> 能匹配 <code>"z"</code> 以及 <code>"zoo"</code> 。 <code>*</code> 等价于 <code>{0,}</code> 。
<b>+</b>	匹配前面的子表达式一次或多次。例如， <code>"zo+"</code> 能匹配 <code>"zo"</code> 以及 <code>"zoo"</code> ，但不能匹配 <code>"z"</code> 。 <code>+</code> 等价于 <code>{1,}</code> 。
<b>?</b>	匹配前面的子表达式零次或一次。例如， <code>"do(es)?"</code> 可以匹配 <code>"does"</code> 或 <code>"does"</code> 中的 <code>"do"</code> 。 <code>?</code> 等价于 <code>{0,1}</code> 。
<b>{n}</b>	<code>n</code> 是一个非负整数。匹配确定的 <code>n</code> 次。例如， <code>"o{2}"</code> 不能匹配 <code>"Bob"</code> 中的 <code>"o"</code> ，但是能匹配 <code>"food"</code> 中的两个 <code>o</code> 。
<b>{n,}</b>	<code>n</code> 是一个非负整数。至少匹配 <code>n</code> 次。例如， <code>"o{2,}"</code> 不能匹配 <code>"Bob"</code> 中的 <code>"o"</code> ，但能匹配 <code>"foooooo"</code> 中的所有 <code>o</code> 。 <code>"o{1,}"</code> 等价于 <code>"o+"</code> 。 <code>"o{0,}"</code> 则等价于 <code>"o*"</code> 。
<b>{n,m}</b>	<code>m</code> 和 <code>n</code> 均为非负整数，其中 <code>n&lt;=m</code> 。最少匹配 <code>n</code> 次且最多匹配 <code>m</code> 次。例如， <code>"o{1,3}"</code> 将匹配 <code>"foooooo"</code> 中的前三个 <code>o</code> 。 <code>"o{0,1}"</code> 等价于 <code>"o?"</code> 。请注意在逗号和两个数之间不能有空格。
<b>?</b>	当该字符紧跟在任何一个其他限制符 ( <code>*</code> , <code>+</code> , <code>?</code> , <code>{n}</code> , <code>{n,}</code> , <code>{n,m}</code> ) 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 <code>"oooo"</code> ， <code>"o+?"</code> 将匹配单个 <code>"o"</code> ，而 <code>"o+"</code> 将匹配所有 <code>"o"</code> 。
<b>.</b>	匹配除 <code>"\n"</code> 之外的任何单个字符。要匹配包括 <code>"\n"</code> 在内的任何字符，请使用像 <code>"(. \n)"</code> 的模式。
<b>(pattern)</b>	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 <code>\$0...\$9</code> 属性。要匹配圆括号字符，请使用 <code>"\"</code> 或 <code>"\"</code> 。
<b>(?:pattern)</b>	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 <code>"( )"</code> 来组合一个模式的各个部分是很有用。例如 <code>"industr(?:y ies)"</code> 就是一个比 <code>"industry industries"</code> 更简略的表达式。
<b>(?=pattern)</b>	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， <code>"Windows(?=95 98 NT 2000)"</code> 能匹配 <code>"Windows2000"</code> 中的 <code>"Windows"</code> ，但不能匹配 <code>"Windows3.1"</code> 中的 <code>"Windows"</code> 。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。
<b>(?!pattern)</b>	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”。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
<b>(?&lt;=pattern)</b>	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”。
<b>(?&lt;!pattern)</b>	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”。
<b>x y</b>	匹配x或y。例如，“z food” 能匹配 “z” 或 “food”。“(z f)ood” 则匹配 “zood” 或 “food”。
<b>[xyz]</b>	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”。
<b>[^xyz]</b>	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 “p”。
<b>[a-z]</b>	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。
<b>[^a-z]</b>	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。
<b>\b</b>	匹配一个单词边界，也就是指单词和空格间的位置。例如，“er\b” 可以匹配 “never” 中的 “er”，但不能匹配 “verb” 中的 “er”。
<b>\B</b>	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”。
<b>\cx</b>	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。
<b>\d</b>	匹配一个数字字符。等价于[0-9]。
<b>\D</b>	匹配一个非数字字符。等价于[^0-9]。
<b>\f</b>	匹配一个换页符。等价于\x0c和\cL。
<b>\n</b>	匹配一个换行符。等价于\x0a和\cJ。
<b>\r</b>	匹配一个回车符。等价于\x0d和\cM。
<b>\s</b>	匹配任何空白字符，包括空格、制表符、换页符等等。等价于[ \f\n\r\t\v]。
<b>\S</b>	匹配任何非空白字符。等价于[^ \f\n\r\t\v]。
<b>\t</b>	匹配一个制表符。等价于\x09和\cI。
<b>\v</b>	匹配一个垂直制表符。等价于\x0b和\cK。
<b>\w</b>	匹配包括下划线的任何单词字符。等价于 “[A-Za-z0-9_]”。

<b>\W</b>	匹配任何非单词字符。等价于 “[^A-Za-z0-9_]”。
<b>\xn</b>	匹配 $n$ ，其中 $n$ 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，“\x41”匹配“A”。“\x041”则等价于“\x04&1”。正则表达式中可以使用ASCII编码。
<b>\num</b>	匹配 $num$ ，其中 $num$ 是一个正整数。对所获取的匹配的引用。例如，“(.)\1”匹配两个连续的不同字符。
<b>\n</b>	标识一个八进制转义值或一个向后引用。如果\ $n$ 之前至少 $n$ 个获取的子表达式，则 $n$ 为向后引用。否则，如果 $n$ 为八进制数字（0-7），则 $n$ 为一个八进制转义值。
<b>\nm</b>	标识一个八进制转义值或一个向后引用。如果\ $nm$ 之前至少有 $nm$ 个获得子表达式，则 $nm$ 为向后引用。如果\ $nm$ 之前至少有 $n$ 个获取，则 $n$ 为一个后跟文字 $m$ 的向后引用。如果前面的条件都不满足，若 $n$ 和 $m$ 均为八进制数字（0-7），则\ $nm$ 将匹配八进制转义值 $nm$ 。
<b>\nml</b>	如果 $n$ 为八进制数字（0-3），且 $m$ 和 $l$ 均为八进制数字（0-7），则匹配八进制转义值 $nml$ 。
<b>\un</b>	匹配 $n$ ，其中 $n$ 是一个用四个十六进制数字表示的Unicode字符。例如，\u00A9匹配版权符号（©）。

## 常用正则表达式

<b>用户名</b>	/^[a-z0-9_-]{3,16}\$/
<b>密码</b>	/^[a-z0-9_-]{6,18}\$/
<b>十六进制值</b>	/^#?([a-f0-9]{6})?([a-f0-9]{3})?\$/
<b>电子邮箱</b>	/^([a-z0-9_\.-]+)@([\da-z\.-]+)\.([a-z\.-]{2,6})\$/ /^[a-z\d]+(\.[a-z\d]+)*@([\da-z](-[\da-z])?)+(\.{1,2}[a-z]+)+\$/
<b>URL</b>	/^(https?:\V)?([\da-z\.-]+)\.([a-z\.-]{2,6})([\Vw \.-]*)\V?\$/
<b>IP 地址</b>	/((2[0-4]\d 25[0-5] [01]?\d\d?)\.){3}(2[0-4]\d 25[0-5] [01]?\d\d?)/ /^(?:(?:25[0-5] 2[0-4][0-9] [01]?[0-9][0-9]?)\.){3}(?:25[0-5] 2[0-4][0-9] [01]?[0-9][0-9]?)\$/
<b>HTML 标签</b>	/^<([a-z]+)([^\<]+)*(?:>(.*)<\1> s+\V>)\$/
<b>删除代码\注释</b>	(?<!http:\S)//.*\$
<b>Unicode 编码中的汉字范围</b>	/^[\u2E80-\u9FFF]+\$/