正则表达式用法大全

一、常见用法

- 只能输入数字："^[0-9]*$"。
- 只能输入 n 位的数字："^\d{n}$"。
- 只能输入至少 n 位的数字："^\d{n,}$"。
- 只能输入 m~n 位的数字：。"^\d{m,n}$"
- 只能输入零和非零开头的数字："^(0|[1-9][0-9]*)$"。
- 只能输入有两位小数的正实数："^[0-9]+(.[0-9]{2})?$"。
- 只能输入有 1~3 位小数的正实数："^[0-9]+(.[0-9]{1,3})?$"。
- 只能输入非零的正整数："^\+?[1-9][0-9]*$"。
- 只能输入非零的负整数："^\-[1-9][]0-9"*$。
- 只能输入长度为 3 的字符："^.{3}$"。
- 只能输入由 26 个英文字母组成的字符串："^[A-Za-z]+$"。
- 只能输入由 26 个大写英文字母组成的字符串："^[A-Z]+$"。
- 只能输入由 26 个小写英文字母组成的字符串："^[a-z]+$"。
- 只能输入由数字和 26 个英文字母组成的字符串："^[A-Za-z0-9]+$"。
- 只能输入由数字、26 个英文字母或者下划线组成的字符串："^\w+$"。
- 验证用户密码："^[a-zA-Z]\w{5,17}$"正确格式为：以字母开头，长度在 6~18 之间，只能包含字符、数字和下划线。
- 验证是否含有^%&',;=?$\"等字符："[^%&',;=?$\x22]+"。
- 只能输入汉字："^[\u4e00-\u9fa5]{0,}$"
- 验证 Email 地址："^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*$"。
- 验证 InternetURL："^http://([\w-]+\.)+[\w-]+(/[\w-./?%&=]*)?$"。
- 验证电话号码："^(\(\d{3,4}-)|\d{3.4}-)?\d{7,8}$"正确格式为："XXX-XXXXXXX"、"XXXX-XXXXXXXX"、"XXX-XXXXXXX"、"XXX-XXXXXXXX"、"XXXXXXX"和"XXXXXXXX"。
- 验证身份证号（15 位或 18 位数字）："^\d{15}|\d{18}$"。
- 验证一年的 12 个月："^(0?[1-9]|1[0-2])$"正确格式为："01"～"09"和"1"～"12"。
- 验证一个月的 31 天："^((0?[1-9])|((1|2)[0-9])|30|31)$"正确格式为；"01"～"09"和"1"～"31"。
- 得用正则表达式从 URL 地址中提取文件名的 javascript 程序，如下结果为 page1

```
s="http://hi.baidu.com/accpandsvse"
s=s.replace(/(.*\/){0,}([^\.]+).*/ig,"$2")
alert(s)
```

- 匹配双字节字符(包括汉字在内)：[^\x00-\xff]

应用：计算字符串的长度（一个双字节字符长度计 2，ASCII 字符计 1）

- String.prototype.len=function(){return this.replace([^\x00-\xff]/g,"aa").length;}
- 匹配空行的正则表达式：\n[\s|]*\r
- 匹配 HTML 标记的正则表达式：/<(.*)>.*<\/\1>|<(.*) \/>/
- 匹配首尾空格的正则表达式：(^\s*)|(\s*$)

```
String.prototype.trim = function()
{
    return this.replace(/(^\s*)|(\s*$)/g, "");
}
```

◆ 利用正则表达式分解和转换 IP 地址：

下面是利用正则表达式匹配 IP 地址，并将 IP 地址转换成对应数值的 Javascript 程序：

```
function IP2V(ip)
{
    re=/(\d+)\.(\d+)\.(\d+)\.(\d+)/g //匹配 IP 地址的正则表达式
    if(re.test(ip))
    {
        return RegExp.$1*Math.pow(255,3))+RegExp.$2*Math.pow(255,2))+RegExp.$3*255+RegExp.$4*1
    }
    else
    {
        throw new Error( "Not a valid IP address!")
    }
}
```

不过上面的程序如果不用正则表达式，而直接用 split 函数来分解可能更简单，程序如下：

```
var ip="10.100.20.168"
ip=ip.split(".")
alert("IP 值是："+(ip[0]*255*255*255+ip[1]*255*255+ip[2]*255+ip[3]*1))
```

二、正则表达式符号解释：

\

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' 匹配字符 "n"。'\n' 匹配一个换行符。序列 '\\' 匹配 "\" 而 "\(" 则匹配 "("。

^

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零次或多次。例如，zo* 能匹配 "z" 以及 "zoo"。* 等价于{0,}。

+

匹配前面的子表达式一次或多次。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 "foooood" 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 "fooooood" 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'。

.

匹配除 "\n" 之外的任何单个字符。要匹配包括 '\n' 在内的任何字符，请使用象 '[.\n]' 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 "Windows 2000" 中的 "Windows" ，但不能匹配 "Windows 3.1" 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。

(?!pattern)

负向预查，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 "Windows 3.1" 中的 "Windows"，但不能匹配 "Windows 2000" 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

x|y

匹配 x 或 y。例如，’z|food’ 能匹配 "z" 或 "food"。’(z|f)ood’ 则匹配 "zood" 或 "food"。

[xyz]

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’。

[^xyz]

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’。

[a-z]

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。

[^a-z]

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

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

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’。

\cx

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。

\d

匹配一个数字字符。等价于 [0-9]。

\D

匹配一个非数字字符。等价于 [^0-9]。

\f

匹配一个换页符。等价于 \x0c 和 \cL。

\n

匹配一个换行符。等价于 \x0a 和 \cJ。

\r

匹配一个回车符。等价于 \x0d 和 \cM。

\s

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]。

\S

匹配任何非空白字符。等价于 [^ \f\n\r\t\v]。

\t

匹配一个制表符。等价于 \x09 和 \cI。

\v

匹配一个垂直制表符。等价于 \x0b 和 \cK。

\w

匹配包括下划线的任何单词字符。等价于'[A-Za-z0-9_]'。

\W

匹配任何非单词字符。等价于 '[^A-Za-z0-9_]'。

\xn

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。.

\num

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' 匹配两个连续的相同字符。

\n

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。

\nm

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。

\nml

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。

\un

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 (?)。