常用正则表达式

"^\d+$"　　//非负整数（正整数 ＋0）
"^[0-9]*[1-9][0-9]*$"　　//正整数
"^((-\d+)|(0+))$"　　//非正整数（负整数 ＋0）
"^-[0-9]*[1-9][0-9]*$"　　//负整数
"^-?\d+$"　　//整数
"^\d+(\.\d+)?$"　　//非负浮点数（正浮点数 ＋0）
"^(([0-9]+\.[0-9]*[1-9][0-9]*)|([0-9]*[1-9][0-9]*\.[0-9]+)|([0-9]*[1-9][0-9]*))$"　　//正浮点数
"^((-\d+(\.\d+)?)|(0+(\.0+)?))$"　　//非正浮点数（负浮点数 ＋0）
"^(-(([0-9]+\.[0-9]*[1-9][0-9]*)|([0-9]*[1-9][0-9]*\.[0-9]+)|([0-9]*[1-9][0-9]*)))$"　　//负浮点数
"^(-?\d+)(\.\d+)?$"　　//浮点数
"^[A-Za-z]+$"　　//由26个英文字母组成的字符串
"^[A-Z]+$"　　//由26个英文字母的大写组成的字符串
"^[a-z]+$"　　//由26个英文字母的小写组成的字符串
"^[A-Za-z0-9]+$"　　//由数字和26个英文字母组成的字符串
"^\w+$"　　//由数字、26个英文字母或者下划线组成的字符串
"^[\w-]+(\.[\w-]+)*@[\w-]+(\.[\w-]+)+$"　　//email地址
"^[a-zA-z]+://(\w+(-\w+)*)(\.(\w+(-\w+)*))*(\?\S*)?$"　　//url
/^(d{2}|d{4})-((0([1-9]{1}))|(1[1|2]))-(([0-2]([1-9]{1}))|(3[0|1]))$/　　//　年-月-日
/^((0([1-9]{1}))|(1[1|2]))/(([0-2]([1-9]{1}))|(3[0|1]))/(d{2}|d{4})$/　　//　月/日/年
"^([w-.]+)@(([[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.)|(([w-]+.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(]?)$"
//Emil
"(d+-)?(d{4}-?d{7}|d{3}-?d{8}|^d{7,8})(-d+)?"　　//电话号码
"^(d{1,2}|1dd|2[0-4]d|25[0-5]).(d{1,2}|1dd|2[0-4]d|25[0-5]).(d{1,2}|1dd|2[0-4]d|25[0-5]).(d{1,2}|1dd|2[0-4]d|25[0-5])$"　　//IP地址

匹配中文字符的正则表达式：　[\u4e00-\u9fa5]

匹配双字节字符(包括汉字在内)：[^\x00-\xff]
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*$)
匹配Email地址的正则表达式：\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*
匹配网址URL的正则表达式：^[a-zA-z]+://(\\w+(-\\w+)*)(\\.(\\w+(-\\w+)*))*(\\?\\S*)?$
匹配帐号是否合法(字母开头，允许5-16字节，允许字母数字下划线)：^[a-zA-Z][a-zA-Z0-9_]{4,15}$
匹配国内电话号码：(\d{3}-|\d{4}-)?(\d{8}|\d{7})?

匹配腾讯QQ号：^[1-9]*[1-9][0-9]*$

下表是元字符及其在正则表达式上下文中的行为的一个完整列表：

\ 将下一个字符标记为一个特殊字符、或一个原义字符、或一个后向引用、或一个八进制转义符。

^ 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。

+ 匹配前面的子表达式一次或多次。+ 等价于 {1,}。

? 匹配前面的子表达式零次或一次。? 等价于 {0,1}。

{n} n 是一个非负整数，匹配确定的n 次。

{n,} n 是一个非负整数，至少匹配n 次。

{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。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。

. 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。

(?: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。

(?=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。

(?!pattern) 负向预查，与**(?=pattern)**作用相反

x|y 匹配 x 或 y。

[xyz] 字符集合。

[^xyz] 负值字符集合。

[a-z] 字符范围，匹配指定范围内的任意字符。

[^a-z] 负值字符范围，匹配任何不在指定范围内的任意字符。

\b 匹配一个单词边界，也就是指单词和空格间的位置。

\B 匹配非单词边界。

\cx 匹配由x指明的控制字符。

\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。

\num 匹配 num，其中num是一个正整数。对所获取的匹配的引用。

\n 标识一个八进制转义值或一个后向引用。如果 \n 之前至少 n 个获取的子表达式，则 n 为后向引用。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后向引用。如果 \nm 之前至少有is preceded by at least nm 个获取得子表达式，则 nm 为后向引用。如果 \nm 之前至少有 n 个获取，则 n 为一个后跟文字 m 的后向引用。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。

匹配中文字符的正则表达式： [u4e00-u9fa5]

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*$)

应用：javascript中没有像vbscript那样的trim函数，我们就可以利用这个表达式来实现，如下：

```
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))
```

匹配Email地址的正则表达式：w+([-+.]w+)*@w+([-.]w+)*.w+([-.]w+)*

匹配网址URL的正则表达式：http://([w-]+.)+[w-]+(/[w- ./?%&=]*)?

利用正则表达式去除字串中重复的字符的算法程序：

```
var s="abacabefgeeii"

var s1=s.replace(/(.).*1/g,"$1")
var re=new RegExp("["+s1+"]","g")
var s2=s.replace(re,"")
alert(s1+s2) //结果为：abcefgi
```

我原来在CSDN上发贴寻求一个表达式来实现去除重复字符的方法，最终没有找到，这是我能想到的最简单的实现方法。思路是使用后向引用取出包括重复的字符，再以重复的字符建立第二个表达式，取到不重复的字符，两者串连。这个方法对于字符顺序有要求的字符串可能不适用。

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://www.9499.net/page1.htm"

s=s.replace(/(.*/){0,}([^.]+).*/ig,"$2")
alert(s)
```

利用正则表达式限制网页表单里的文本框输入内容：

用正则表达式限制只能输入中文：onkeyup="value=value.replace(/[^u4E00-u9FA5]/g,'')" onbeforepaste="clipboardData.setData('text',clipboardData.getData('text').replace(/[^u4E00-u9FA5]/g,''))"

用正则表达式限制只能输入全角字符：onkeyup="value=value.replace(/[^uFF00-uFFFF]/g,'')" onbeforepaste="clipboardData.setData('text',clipboardData.getData('text').replace(/[^uFF00-uFFFF]/g,''))"

用正则表达式限制只能输入数字：onkeyup="value=value.replace(/[^d]/g,'') "onbeforepaste="clipboardData.setData('text',clipboardData.getData('text').replace(/[^d]/g,''))"

用正则表达式限制只能输入数字和英文：onkeyup="value=value.replace(/[W]/g,'') "onbeforepaste="clipboardData.setData('text',clipboardData.getData('text').replace(/[^d]/g,''))"