正则表达式总结

作者：[lmb55](http://my.csdn.net/lmb55)

每次使用到正则表达式的时候都要去翻书或者去网上查，很麻烦，抽出点时间做一个总结，方便以后用到的时候查阅。

一、正则表达式的几个特殊符号

1、^和$

两个特殊的符号‘^’和’$’。他们的作用是分别指出一个字符串的开始和结束。

例子如下：

"^The"：表示所有以"The"开始的字符串（"There"，"The cat"等）；

"of despair$"：表示所以以"of despair"结尾的字符串；

"^abc$"：表示开始和结尾都是"abc"的字符串——呵呵，只有"abc"自己了；

"notice"：表示任何包含"notice"的字符串。

最后那个例子，如果你不使用两个特殊字符，你就在表示要查找的串在被查找串的任意部分——你并不把它定位在某一个顶端。

2、\*，+和?

其它还有‘\*’，’+’和’?’这三个符号，表示一个或一序列字符重复出现的次数。它们分别表示“没有或更多”，“一次或更多”还有“没有或一次”。

下面是几个例子：

"ab\*"：表示一个字符串有一个a后面跟着零个或若干个b。（"a", "ab", "abbb",……）；

"ab+"：表示一个字符串有一个a后面跟着至少一个b或者更多；

"ab?"：表示一个字符串有一个a后面跟着零个或者一个b；

"a?b+$"：表示在字符串的末尾有零个或一个a跟着一个或几个b。

3、大括号{}

你也可以使用范围，用大括号括起，用以表示重复次数的范围。

示例如下：

"ab{2}"：表示一个字符串有一个a跟着2个b（"abb"）；

"ab{2,}"：表示一个字符串有一个a跟着至少2个b；

"ab{3,5}"：表示一个字符串有一个a跟着3到5个b。

请注意，你必须指定范围的下限（如："{0,2}"而不是"{,2}"）。还有，你可能注意到了，'\*'，'+'和'?'相当于"{0,}"，"{1,}"和"{0,1}"。

4、¦

还有一个’¦‘，表示“或”操作：

示例如下：

"hi¦hello"：表示一个字符串里有"hi"或者"hello"；

"(b¦cd)ef"：表示"bef"或"cdef"；

"(a¦b)\*c"：表示一串"a""b"混合的字符串后面跟一个"c"；

5、.

‘.’可以替代任何字符：

示例如下：

"a.[0-9]"：表示一个字符串有一个"a"后面跟着一个任意字符和一个数字；

"^.{3}$"：表示有任意三个字符的字符串（长度为3个字符）；

6、方括号[ ]

方括号表示某些字符允许在一个字符串中的某一特定位置出现：

示例如下：

"[ab]"：表示一个字符串有一个"a"或"b"（相当于"a¦b"）；

"[a-d]"：表示一个字符串包含小写的'a'到'd'中的一个（相当于"a¦b¦c¦d"或者"[abcd]"）；

"^[a-zA-Z]"：表示一个以字母开头的字符串；

"[0-9]%"：表示一个百分号前有一位的数字；

",[a-zA-Z0-9]$"：表示一个字符串以一个逗号后面跟着一个字母或数字结束。

你也可以在方括号里用'^'表示不希望出现的字符，'^'应在方括号里的第一位。（如："%[^a-zA-Z]%"表示两个百分号中不应该出现字母）。

为了逐字表达，你必须在"^.$()¦\*+?{\"这些字符前加上转移字符'\'。

请注意在方括号中，不需要转义字符。

注意：

^ 表示打头的字符要匹配紧跟^后面的规则

$ 表示打头的字符要匹配紧靠$前面的规则

[ ] 中的内容是可选字符集

[0-9] 表示要求字符范围在0-9之间

{1,20}表示数字字符串长度合法为1到20，即为[0-9]中的字符出现次数的范围是1到20次。

/^ 和 $/成对使用应该是表示要求整个字符串完全匹配定义的规则，而不>是只匹配字符串中的一个子串。

二、正则表达式应用实例

校验是否全由数字组成

/^[0-9]{1,20}$/

校验登录名：只能输入5-20个以字母开头、可带数字、“\_”、“.”的字串

/^[a-zA-Z]{1}([a-zA-Z0-9]|[.\_]){4,19}$/

^[a-zA-Z]{1} 表示第一个字符要求是字母。

([a-zA-Z0-9]|[.\_]){4,19}表示从第二位开始（因为它紧跟在上个表达式后面）的一个长度为4到9位的字符串，它要求是由大小写字母、数字或者特殊字符集[.\_]组成。

校验用户姓名：只能输入1-30个以字母开头的字串

/^[a-zA-Z]{1,30}$/

校验密码：只能输入6-20个字母、数字、下划线

/^(\w){6,20}$/

\w：用于匹配字母，数字或下划线字符

校验普通电话、传真号码：可以“+”或数字开头，可含有“-” 和 “ ”

/^[+]{0,1}(\d){1,3}[ ]?([-]?((\d)|[ ]){1,12})+$/

\d：用于匹配从0到9的数字；

“?”元字符规定其前导对象必须在目标对象中连续出现零次或一次

可以匹配的字符串如：+123 -999 999 ； +123-999 999 ；123 999 999 ；+123 999999等

校验URL

/^http[s]{0,1}:\/\/.+$/ 或 /^http[s]{0,1}:\/\/.{1,n}$/ (表示url串的长度为length(“https://”) + n )

\ /表示字符“/”。

. 表示所有字符的集

+ 等同于{1,}，就是1到正无穷吧。

校验纯中文字符

/^[\u4E00-\u9FA5]+$/

[\u4E00-\u9FA5] ：匹配中文字符

三、常用正则表达式大全

匹配中文字符的正则表达式： [u4e00-u9fa5]

　　评注：匹配中文还真是个头疼的事，有了这个表达式就好办了

　　匹配双字节字符(包括汉字在内)：[^x00-xff]

　　评注：可以用来计算字符串的长度（一个双字节字符长度计2，ASCII字符计1）

　　匹配空白行的正则表达式：ns\*r

　　评注：可以用来删除空白行

　　匹配HTML标记的正则表达式：<(S\*?)[^>]\*>.\*?|<.\*? />

　　匹配首尾空白字符的正则表达式：^s\*|s\*$

　　评注：可以用来删除行首行尾的空白字符(包括空格、制表符、换页符等等)，非常有用的表达式

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

　　评注：表单验证时很实用

　　匹配网址URL的正则表达式：[a-zA-z]+://[^s]\*

　　评注：网上流传的版本功能很有限，上面这个基本可以满足需求

　　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}$

　　评注：表单验证时很实用

　　匹配国内电话号码：d{3}-d{8}|d{4}-d{7}

　　评注：匹配形式如 0511-4405222 或 021-87888822

　　匹配腾讯QQ号：[1-9][0-9]{4,}

　　评注：腾讯QQ号从10000开始

　　匹配中国邮政编码：[1-9]d{5}(?!d)

　　评注：中国邮政编码为6位数字

　　匹配身份证：d{15}|d{18}

　　评注：中国的身份证为15位或18位

　　匹配ip地址：d+.d+.d+.d+

　　评注：提取ip地址时有用

　　匹配特定数字：

　　^[1-9]d\*$　 　 //匹配正整数

　　^-[1-9]d\*$ 　 //匹配负整数

　　^-?[1-9]d\*$　　 //匹配整数

　　^[1-9]d\*|0$　 //匹配非负整数（正整数 + 0）

　　^-[1-9]d\*|0$　　 //匹配非正整数（负整数 + 0）

　　^[1-9]d\*.d\*|0.d\*[1-9]d\*$　　 //匹配正浮点数

　　^-([1-9]d\*.d\*|0.d\*[1-9]d\*)$　 //匹配负浮点数

　　^-?([1-9]d\*.d\*|0.d\*[1-9]d\*|0?.0+|0)$　 //匹配浮点数

　　^[1-9]d\*.d\*|0.d\*[1-9]d\*|0?.0+|0$　　 //匹配非负浮点数（正浮点数 + 0）

　　^(-([1-9]d\*.d\*|0.d\*[1-9]d\*))|0?.0+|0$　　//匹配非正浮点数（负浮点数 + 0）

　　评注：处理大量数据时有用，具体应用时注意修正

　　匹配特定字符串：

　　^[A-Za-z]+$　　//匹配由26个英文字母组成的字符串

　　^[A-Z]+$　　//匹配由26个英文字母的大写组成的字符串

　　^[a-z]+$　　//匹配由26个英文字母的小写组成的字符串

　　^[A-Za-z0-9]+$　　//匹配由数字和26个英文字母组成的字符串

　　^w+$　　//匹配由数字、26个英文字母或者下划线组成的字符串

　　在使用RegularExpressionValidator验证控件时的验证功能及其验证表达式介绍如下:

　　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：“XXXX-XXXXXXX”，“XXXX-XXXXXXXX”，“XXX-XXXXXXX”，

　　“XXX-XXXXXXXX”，“XXXXXXX”，“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”。

　　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：/<(.\*)>.\*|<(.\*) />/

　　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：http://([w-]+.)+[w-]+(/[w- ./?%&=]\*)?

　　(1)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;}

　　(2)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

　　String.prototype.trim = function()

　　{

　　return this.replace(/(^s\*)|(s\*$)/g, "");

　　}

　　(3)应用：利用正则表达式分解和转换IP地址

　　function IP2V(ip) //IP地址转换成对应数值

　　{

　　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!")

　　}

　　}

　　(4)应用：从URL地址中提取文件名的javascript程序

　　s="http://www.9499.net/page1.htm";

　　s=s.replace(/(.\*/){0,}([^.]+).\*/ig,"$2") ;//Page1.htm

　　(5)应用：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,''

四、javascript正则表达式检验

<**script**>  
 *//校验是否全由数字组成* **function** *isDigit*(s) {  
 **var** patrn=/^[0-9]{1,20}$/;  
 **if** (!patrn.exec(s)) **return false  
 return true** }  
  
 *//校验登录名：只能输入5-20个以字母开头、可带数字、“\_”、“.”的字串* **function** *isRegisterUserName*(s) {  
 **var** patrn=/^[a-zA-Z]{1}([a-zA-Z0-9]|[.\_]){4,19}$/;  
 **if** (!patrn.exec(s)) **return false  
 return true** }  
  
 *//校验用户姓名：只能输入1-30个以字母开头的字串* **function** *isTrueName*(s) {  
 **var** patrn=/^[a-zA-Z]{1,30}$/;  
 **if** (!patrn.exec(s)) **return false  
 return true** }  
  
 *//校验密码：只能输入6-20个字母、数字、下划线* **function** *isPasswd*(s) {  
 **var** patrn=/^(\w){6,20}$/;  
 **if** (!patrn.exec(s)) **return false  
 return true** }  
  
 *//校验普通电话、传真号码：可以“+”开头，除数字外，可含有“-”* **function** *isTel*(s) {  
 *//var patrn=/^[+]{0,1}(\d){1,3}[ ]?([-]?(\d){1,12})+$/;* **var** patrn=/^[+]{0,1}(\d){1,3}[ ]?([-]?((\d)|[ ]){1,12})+$/;  
 **if** (!patrn.exec(s)) **return false  
 return true** }  
  
 *//校验手机号码：必须以数字开头，除数字外，可含有“-”* **function** *isMobil*(s) {  
 **var** patrn=/^[+]{0,1}(\d){1,3}[ ]?([-]?((\d)|[ ]){1,12})+$/;  
 **if** (!patrn.exec(s)) **return false  
 return true** }  
  
 *//校验邮政编码* **function** *isPostalCode*(s) {  
*//var patrn=/^[a-zA-Z0-9]{3,12}$/;* **var** patrn=/^[a-zA-Z0-9 ]{3,12}$/;  
 **if** (!patrn.exec(s)) **return false  
 return true** }  
  
 *//校验搜索关键字* **function** *isSearch*(s) {  
 **var** patrn=/^[^`~!@#$%^&\*()+=|\\\][\]\{\}:;\'\,.<>/?]{1}[^`~!@$%^&()+=|\\\][\]\{\}:;\'\,.<>?]{0,19}$/;  
 **if** (!patrn.exec(s)) **return false  
 return true** }  
  
 **function** *isIP*(s) *//by zergling{* **var *patrn***=/^[0-9.]{1,20}$/;  
 **if** (!***patrn***.exec(s)) **return false  
 return true** }  
  
  
 *//验证val是（true）否(false)在lo和hi之间* **function** *isBetween* (val, lo, hi) {  
 **if** ((val < lo) || (val > hi)) { **return**(**false**); }  
 **else** { **return**(**true**); }  
 }  
  
 *//检查theStr是（true）否（false）是一个有效的日期* **function** *isDate* (theStr) {  
 **var** the1st = theStr.indexOf(**'-'**);  
 **var** the2nd = theStr.lastIndexOf(**'-'**);  
  
 **if** (the1st == the2nd) { **return**(**false**); }  
 **else** {  
 **var** y = theStr.substring(0,the1st);  
 **var** m = theStr.substring(the1st+1,the2nd);  
 **var** d = theStr.substring(the2nd+1,theStr.**length**);  
 **var** maxDays = 31;  
  
 **if** (*isInt*(m)==**false** || *isInt*(d)==**false** || *isInt*(y)==**false**) {  
 **return**(**false**); }  
 **else if** (y.**length** < 4) { **return**(**false**); }  
 **else if** (!*isBetween* (m, 1, 12)) { **return**(**false**); }  
 **else if** (m==4 || m==6 || m==9 || m==11) maxDays = 30;  
 **else if** (m==2) {  
 **if** (y % 4 > 0) maxDays = 28;  
 **else if** (y % 100 == 0 && y % 400 > 0) maxDays = 28;  
 **else** maxDays = 29;  
 }  
 **if** (*isBetween*(d, 1, maxDays) == **false**) { **return**(**false**); }  
 **else** { **return**(**true**); }  
 }  
 }  
  
 *//检查theStr是否是英文日期格式下的有效日期* **function** *isEuDate* (theStr) {  
 **if** (*isBetween*(theStr.**length**, 8, 10) == **false**) { **return**(**false**); }  
 **else** {  
 **var** the1st = theStr.indexOf(**'/'**);  
 **var** the2nd = theStr.lastIndexOf(**'/'**);  
  
 **if** (the1st == the2nd) { **return**(**false**); }  
 **else** {  
 **var** m = theStr.substring(the1st+1,the2nd);  
 **var** d = theStr.substring(0,the1st);  
 **var** y = theStr.substring(the2nd+1,theStr.**length**);  
 **var** maxDays = 31;  
  
 **if** (*isInt*(m)==**false** || *isInt*(d)==**false** || *isInt*(y)==**false**) {  
 **return**(**false**); }  
 **else if** (y.**length** < 4) { **return**(**false**); }  
 **else if** (*isBetween* (m, 1, 12) == **false**) { **return**(**false**); }  
 **else if** (m==4 || m==6 || m==9 || m==11) maxDays = 30;  
 **else if** (m==2) {  
 **if** (y % 4 > 0) maxDays = 28;  
 **else if** (y % 100 == 0 && y % 400 > 0) maxDays = 28;  
 **else** maxDays = 29;  
 }  
  
 **if** (*isBetween*(d, 1, maxDays) == **false**) { **return**(**false**); }  
 **else** { **return**(**true**); }  
 }  
 }  
  
 }  
  
 *//比较两个日期那个是最新的，如果lessDate<moreDate返回true，否则返回false* **function** *isComdate* (lessDate , moreDate) {  
 **if** (!*isDate*(lessDate)) { **return**(**false**);}  
 **if** (!*isDate*(moreDate)) { **return**(**false**);}  
 **var** less1st = lessDate.indexOf(**'-'**);  
 **var** less2nd = lessDate.lastIndexOf(**'-'**);  
 **var** more1st = moreDate.indexOf(**'-'**);  
 **var** more2nd = moreDate.lastIndexOf(**'-'**);  
 **var** lessy = lessDate.substring(0,less1st);  
 **var** lessm = lessDate.substring(less1st+1,less2nd);  
 **var** lessd = lessDate.substring(less2nd+1,lessDate.**length**);  
 **var** morey = moreDate.substring(0,more1st);  
 **var** morem = moreDate.substring(more1st+1,more2nd);  
 **var** mored = moreDate.substring(more2nd+1,moreDate.**length**);  
 **var** Date1 = **new** Date(lessy,lessm,lessd);  
 **var** Date2 = **new** Date(morey,morem,mored);  
 **if** (Date1>Date2) { **return**(**false**);}  
 **return**(**true**);  
  
 }  
  
 *//检查参数是否为空* **function** *isEmpty* (str) {  
 **if** ((str==**null**)||(str.**length**==0)) **return true**;  
 **else return**(**false**);  
 }  
  
 *//检查参数是否是Integer类型* **function** *isInt* (theStr) {  
 **var** flag = **true**;  
  
 **if** (*isEmpty*(theStr)) { flag=**false**; }  
 **else** { **for** (**var** i=0; i<theStr.**length**; i++) {  
 **if** (*isDigit*(theStr.substring(i,i+1)) == **false**) {  
 flag = **false**; **break**;  
 }  
 }  
 }  
 **return**(flag);  
 }  
  
 */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* FUNCTION: isReal  
 \* PARAMETER: heStr AS String  
 decLen AS Integer (how many digits after period)  
 \* RETURNS: TRUE if theStr is a float, otherwise FALSE  
 \* CALLS: isInt  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* **function** *isReal* (theStr, decLen) {  
 **var** dot1st = theStr.indexOf(**'.'**);  
 **var** dot2nd = theStr.lastIndexOf(**'.'**);  
 **var** OK = **true**;  
  
 **if** (*isEmpty*(theStr)) **return false**;  
  
 **if** (dot1st == -1) {  
 **if** (!*isInt*(theStr)) **return**(**false**);  
 **else return**(**true**);  
 }  
  
 **else if** (dot1st != dot2nd) **return** (**false**);  
 **else if** (dot1st==0) **return** (**false**);  
 **else** {  
 **var** intPart = theStr.substring(0, dot1st);  
 **var** decPart = theStr.substring(dot2nd+1);  
  
 **if** (decPart.**length** > decLen) **return**(**false**);  
 **else if** (!*isInt*(intPart) || !*isInt*(decPart)) **return** (**false**);  
 **else if** (*isEmpty*(decPart)) **return** (**false**);  
 **else return**(**true**);  
 }  
 }  
  
 *//检验参数是否是一个有效的Email地址* **function** *isEmail* (theStr) {  
 **var** atIndex = theStr.indexOf(**'@'**);  
 **var** dotIndex = theStr.indexOf(**'.'**, atIndex);  
 **var** flag = **true**;  
 **theSub** = theStr.substring(0, dotIndex+1)  
  
 **if** ((atIndex < 1)||(atIndex != theStr.lastIndexOf(**'@'**))||(dotIndex < atIndex + 2)||(theStr.**length** <= **theSub**.**length**))  
 { **return**(**false**); }  
 **else** { **return**(**true**); }  
 }  
  
 */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* FUNCTION: newWindow  
 \* PARAMETERS: doc -> Document to open in the new window  
 hite -> Height of the new window  
 wide -> Width of the new window  
 bars -> 1-Scroll bars = YES 0-Scroll Bars = NO  
 resize -> 1-Resizable = YES 0-Resizable = NO  
 \* CALLS: NONE  
 \* RETURNS: New window instance  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* **function** *newWindow* (doc, hite, wide, bars, resize) {  
 **var** winNew=**"\_blank"**;  
 **var** opt=**"toolbar=0,location=0,directories=0,status=0,menubar=0,"**;  
 opt+=(**"scrollbars="**+bars+**","**);  
 opt+=(**"resizable="**+resize+**","**);  
 opt+=(**"width="**+wide+**","**);  
 opt+=(**"height="**+hite);  
 **winHandle**=**window**.open(doc,winNew,opt);  
 **return**;  
 }  
  
 */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* FUNCTION: DecimalFormat  
 \* PARAMETERS: paramValue -> Field value  
 \* CALLS: NONE  
 \* RETURNS: Formated string  
 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/* **function** *DecimalFormat* (paramValue) {  
 **var** intPart = parseInt(paramValue);  
 **var** decPart =parseFloat(paramValue) - intPart;  
  
 **str** = **""**;  
 **if** ((decPart == 0) || (decPart == **null**)) **str** += (intPart + **".00"**);  
 **else str** += (intPart + decPart);  
  
 **return** (**str**);  
 }  
 </**script**>