[正则表达式]文本框输入内容控制

整数或者小数：^[0-9]+\.{0,1}[0-9]{0,2}$

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"。

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标签的正则表达式：<(.\*)>(.\*)<\/(.\*)>|<(.\*)\/>

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,''))"

<input onkeyup="value=value.replace(/[^\u4E00-\u9FA5\w]/g,'')" onbeforepaste="clipboardData.setData('text',clipboardData.getData('text').replace(/[^\u4E00-\u9FA5\w]/g,''))" value="允许下划线,数字字母和汉字">

<script language="javascript">

if (document.layers)//触发键盘事件

document.captureEvents(Event.KEYPRESS)

function xz(thsv,nob){

if(nob=="2"){

window.clipboardData.setData("text","")

alert("避免非法字符输入,请勿复制字符");

return false;

}

if (event.keyCode!=8 && event.keyCode!=16 && event.keyCode!=37 && event.keyCode!=38 && event.keyCode!=39 && event.keyCode!=40){

thsvv=thsv.value;//输入的值

thsvs=thsvv.substring(thsvv.length-1);//输入的最后一个字符

//thsvss=thsvv.substring(0,thsvv.length-1);//去掉最后一个错误字符

if (!thsvs.replace(/[^\u4E00-\u9FA5\w]/g,'') || event.keyCode==189){//正则除去符号和下划线 key

thsv.value='请勿输入非法符号 ['+thsvs+']';

alert('请勿输入非法符号 ['+thsvs+']');

thsv.value="";

return false;

}

}

}

</script>

<input onkeyup="xz(this,1)" onPaste="xz(this,2)" value="">允许数字字母和汉字

<script language="javascript">

<!--

function MaxLength(field,maxlimit){

var j = field.value.replace(/[^\x00-\xff]/g,"\*\*").length;

//alert(j);

var tempString=field.value;

var tt="";

if(j > maxlimit){

for(var i=0;i<maxlimit;i++){

if(tt.replace(/[^\x00-\xff]/g,"\*\*").length < maxlimit)

tt = tempString.substr(0,i+1);

else

break;

}

if(tt.replace(/[^\x00-\xff]/g,"\*\*").length > maxlimit)

tt=tt.substr(0,tt.length-1);

field.value = tt;

}else{

;

}

}

</script>

单行文本框控制<br />

<INPUT type="text" id="Text1" name="Text1" onpropertychange="MaxLength(this, 5)"><br />

多行文本框控制:<br />

<TEXTAREA rows="14"

cols="39" id="Textarea1" name="Textarea1" onpropertychange="MaxLength(this, 15)"></TEXTAREA><br />

控制表单内容只能输入数字,中文....

<script>

function test()

{

if(document.a.b.value.length>50)

{

alert("不能超过50个字符！");

document.a.b.focus();

return false;

}

}

</script>

<form name=a onsubmit="return test()">

<textarea name="b" cols="40" wrap="VIRTUAL" rows="6"></textarea>

<input type="submit" name="Submit" value="check">

</form>

只能是汉字

<input onkeyup="value=value.replace(/[^\u4E00-\u9FA5]/g,'')">

只能是英文字符

<script language=javascript>

function onlyEng()

{

if(!(event.keyCode>=65&&event.keyCode<=90))

event.returnValue=false;

}

</script>

<input onkeydown="onlyEng();">

<input name="coname" type="text" size="50" maxlength="35" class="input2" onkeyup="value=value.replace(/[\W]/g,'') "onbeforepaste="clipboardData.setData('text',clipboardData.getData('text').replace(/[^\d]/g,''))">

只能是数字

<script language=javascript>

function onlyNum()

{

if(!((event.keyCode>=48&&event.keyCode<=57)||(event.keyCode>=96&&event.keyCode<=105)))

//考虑小键盘上的数字键

event.returnValue=false;

}

</script>

<input onkeydown="onlyNum();">

只能是英文字符和数字

<input onkeyup="value=value.replace(/[\W]/g,'') "onbeforepaste="clipboardData.setData('text',clipboardData.getData('text').replace(/[^\d]/g,''))">

验证为email格式

<SCRIPT LANGUAGE=Javascript RUNAT=Server>

function isEmail(strEmail) {

if (strEmail.search(/^\w+((-\w+)|(\.\w+))\*\@[A-Za-z0-9]+((\.|-)[A-Za-z0-9]+)\*\.[A-Za-z0-9]+$/) != -1)

return true;

else

alert("oh");

}

</SCRIPT>

<input type=text onblur=isEmail(this.value)>

屏蔽关键字(sex , fuck) - 已修改

<script language="JavaScript1.2">

function test() {

if((a.b.value.indexOf ("sex") == 0)||(a.b.value.indexOf ("fuck") == 0)){

alert("五讲四美三热爱");

a.b.focus();

return false;}

}

</script>

<form name=a onsubmit="return test()">

<input type=text name=b>

<input type="submit" name="Submit" value="check">

</form>

限制文本框里只能输入数字

<input onkeyup="if(event.keyCode !=37 && event.keyCode != 39) value=value.replace(/\D/g,'');"onbeforepaste="clipboardData.setData('text',clipboardData.getData('text').replace(/\D/g,''))">

<PIXTEL\_MMI\_EBOOK\_2005>2 </PIXTEL\_MMI\_EBOOK\_2005>

JAVA正则表达式语法（转）

正则表达式语法

正则表达式是一种文本模式，包括普通字符（例如，a 到 z 之间的字母）和特殊字符（称为“元字符”）。模式描述在搜索文本时要匹配的一个或多个字符串。

正则表达式示例

表达式 匹配

/^\s\*$/

匹配空行。

/\d{2}-\d{5}/

验证由两位数字、一个连字符再加 5 位数字组成的 ID 号。

/<\s\*(\S+)(\s[^>]\*)?>[\s\S]\*<\s\*\/\1\s\*>/

匹配 HTML 标记。

下表包含了元字符的完整列表以及它们在正则表达式上下文中的行为：

字符 说明

\

将下一字符标记为特殊字符、文本、反向引用或八进制转义符。例如，“n”匹配字符“n”。“\n”匹配换行符。序列“\\”匹配“\”，“\(”匹配“(”。

^

匹配输入字符串开始的位置。如果设置了 RegExp 对象的 Multiline 属性，^ 还会与“\n”或“\r”之后的位置匹配。

$

匹配输入字符串结尾的位置。如果设置了 RegExp 对象的 Multiline 属性，$ 还会与“\n”或“\r”之前的位置匹配。

\*

零次或多次匹配前面的字符或子表达式。例如，zo\* 匹配“z”和“zoo”。\* 等效于 {0,}。

+

一次或多次匹配前面的字符或子表达式。例如，“zo+”与“zo”和“zoo”匹配，但与“z”不匹配。+ 等效于 {1,}。

?

零次或一次匹配前面的字符或子表达式。例如，“do(es)?”匹配“do”或“does”中的“do”。? 等效于 {0,1}。

{n}

n 是非负整数。正好匹配 n 次。例如，“o{2}”与“Bob”中的“o”不匹配，但与“food”中的两个“o”匹配。

{n,}

n 是非负整数。至少匹配 n 次。例如，“o{2,}”不匹配“Bob”中的“o”，而匹配“foooood”中的所有 o。“o{1,}”等效于“o+”。“o{0,}”等效于“o\*”。

{n,m}

M 和 n 是非负整数，其中 n <= m。匹配至少 n 次，至多 m 次。例如，“o{1,3}”匹配“fooooood”中的头三个 o。'o{0,1}' 等效于 'o?'。注意：您不能将空格插入逗号和数字之间。

?

当此字符紧随任何其他限定符（\*、+、?、{n}、{n,}、{n,m}）之后时，匹配模式是“非贪心的”。“非贪心的”模式匹配搜索到的、尽可能短的字符串，而默认的“贪心的”模式匹配搜索到的、尽可能长的字符串。例如，在字符串“oooo”中，“o+?”只匹配单个“o”，而“o+”匹配所有“o”。

.

匹配除“\n”之外的任何单个字符。若要匹配包括“\n”在内的任意字符，请使用诸如“[\s\S]”之类的模式。

(pattern)

匹配 pattern 并捕获该匹配的子表达式。可以使用 $0…$9 属性从结果“匹配”集合中检索捕获的匹配。若要匹配括号字符 ( )，请使用“\(”或者“\)”。

(?:pattern)

匹配 pattern 但不捕获该匹配的子表达式，即它是一个非捕获匹配，不存储供以后使用的匹配。这对于用“or”字符 (|) 组合模式部件的情况很有用。例如，'industr(?:y|ies) 是比 'industry|industries' 更经济的表达式。

(?=pattern)

执行正向预测先行搜索的子表达式，该表达式匹配处于匹配 pattern 的字符串的起始点的字符串。它是一个非捕获匹配，即不能捕获供以后使用的匹配。例如，'Windows (?=95|98|NT|2000)' 匹配“Windows 2000”中的“Windows”，但不匹配“Windows 3.1”中的“Windows”。预测先行不占用字符，即发生匹配后，下一匹配的搜索紧随上一匹配之后，而不是在组成预测先行的字符后。

(?!pattern)

执行反向预测先行搜索的子表达式，该表达式匹配不处于匹配 pattern 的字符串的起始点的搜索字符串。它是一个非捕获匹配，即不能捕获供以后使用的匹配。例如，'Windows (?!95|98|NT|2000)' 匹配“Windows 3.1”中的 “Windows”，但不匹配“Windows 2000”中的“Windows”。预测先行不占用字符，即发生匹配后，下一匹配的搜索紧随上一匹配之后，而不是在组成预测先行的字符后。

x|y

匹配 x 或 y。例如，'z|food' 匹配“z”或“food”。'(z|f)ood' 匹配“zood”或“food”。

[xyz]

字符集。匹配包含的任一字符。例如，“[abc]”匹配“plain”中的“a”。

[^xyz]

反向字符集。匹配未包含的任何字符。例如，“[^abc]”匹配“plain”中的“p”。

[a-z]

字符范围。匹配指定范围内的任何字符。例如，“[a-z]”匹配“a”到“z”范围内的任何小写字母。

[^a-z]

反向范围字符。匹配不在指定的范围内的任何字符。例如，“[^a-z]”匹配任何不在“a”到“z”范围内的任何字符。

\b

匹配一个字边界，即字与空格间的位置。例如，“er\b”匹配“never”中的“er”，但不匹配“verb”中的“er”。

\B

非字边界匹配。“er\B”匹配“verb”中的“er”，但不匹配“never”中的“er”。

\cx

匹配 x 指示的控制字符。例如，\cM 匹配 Control-M 或回车符。x 的值必须在 A-Z 或 a-z 之间。如果不是这样，则假定 c 就是“c”字符本身。

\d

数字字符匹配。等效于 [0-9]。

\D

非数字字符匹配。等效于 [^0-9]。

\f

换页符匹配。等效于 \x0c 和 \cL。

\n

换行符匹配。等效于 \x0a 和 \cJ。

\r

匹配一个回车符。等效于 \x0d 和 \cM。

\s

匹配任何空白字符，包括空格、制表符、换页符等。与 [ \f\n\r\t\v] 等效。

\S

匹配任何非空白字符。与 [^ \f\n\r\t\v] 等效。

\t

制表符匹配。与 \x09 和 \cI 等效。

\v

垂直制表符匹配。与 \x0b 和 \cK 等效。

\w

匹配任何字类字符，包括下划线。与“[A-Za-z0-9\_]”等效。

\W

与任何非单词字符匹配。与“[^A-Za-z0-9\_]”等效。

\xn

匹配 n，此处的 n 是一个十六进制转义码。十六进制转义码必须正好是两位数长。例如，“\x41”匹配“A”。“\x041”与“\x04”&“1”等效。允许在正则表达式中使用 ASCII 代码。

\num

匹配 num，此处的 num 是一个正整数。到捕获匹配的反向引用。例如，“(.)\1”匹配两个连续的相同字符。

\n

标识一个八进制转义码或反向引用。如果 \n 前面至少有 n 个捕获子表达式，那么 n 是反向引用。否则，如果 n 是八进制数 (0-7)，那么 n 是八进制转义码。

\nm

标识一个八进制转义码或反向引用。如果 \nm 前面至少有 nm 个捕获子表达式，那么 nm 是反向引用。如果 \nm 前面至少有 n 个捕获，则 n 是反向引用，后面跟有字符 m。如果两种前面的情况都不存在，则 \nm 匹配八进制值 nm，其中 n 和 m 是八进制数字 (0-7)。

\nml

当 n 是八进制数 (0-3)，m 和 l 是八进制数 (0-7) 时，匹配八进制转义码 nml。

\un

匹配 n，其中 n 是以四位十六进制数表示的 Unicode 字符。例如，\u00A9 匹配版权符号 (?)。