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,''))"

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：\n\s\*\r

评注：可以用来删除空白行

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

评注：网上流传的版本太糟糕，上面这个也仅仅能匹配部分，对于复杂的嵌套标记依旧无能为力

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

评注：最基本也是最常用的一些表达式

整理出来的一些常用的正则表达式所属分类: JScript  (三)

Email : /^\w+([-+.]\w+)\*@\w+([-.]\\w+)\*\.\w+([-.]\w+)\*$/

isEmail1 : /^\w+([\.\-]\w+)\*\@\w+([\.\-]\w+)\*\.\w+$/;

isEmail2 : /^.\*@[^\_]\*$/;

Phone : /^((\(\d{3}\))|(\d{3}\-))?(\(0\d{2,3}\)|0\d{2,3}-)?[1-9]\d{6,7}$/

Mobile : /^((\(\d{3}\))|(\d{3}\-))?13\d{9}$/

Url : /^http:\/\/[A-Za-z0-9]+\.[A-Za-z0-9]+[\/=\?%\-&\_~`@[\]\':+!]\*([^<>\"\"])\*$/

IdCard : /^\d{15}(\d{2}[A-Za-z0-9])?$/

Currency : /^\d+(\.\d+)?$/

Number : /^\d+$/

Code : /^[1-9]\d{5}$/

QQ : /^[1-9]\d{4,8}$/

Integer : /^[-\+]?\d+$/

Double : /^[-\+]?\d+(\.\d+)?$/

English : /^[A-Za-z]+$/

Chinese : /^[\u0391-\uFFE5]+$/

UnSafe : /^(([A-Z]\*|[a-z]\*|\d\*|[-\_\~!@#\$%\^&\\*\.\(\)\[\]\{\}<>\?\\\/\'\"]\*)|.{0,5})$|\s/

PassWord :^[\\w]{6,12}$

ZipCode : ^[\\d]{6}

/^(\+\d+ )?(\(\d+\) )?[\d ]+$/; //这个是国际通用的电话号码判断

/^(1[0-2]\d|\d{1,2})$/; //这个是年龄的判断

/^\d+\.\d{2}$/;  //这个是判断输入的是否为货币值

 <!-- IP地址有效性验证函数-->

 <script language=javascript runat=server>

    ip\_ip = '(25[0-5]|2[0-4]\\d|1\\d\\d|\\d\\d|\\d)';

    ip\_ipdot = ip + '\\.';

    isIPaddress = new RegExp('^'+ip\_ipdot+ip\_ipdot+ipdot+ip\_ip+'$');

</script>

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;}

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, "");

}

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

^\d+$　　//匹配非负整数（正整数 + 0）

^[0-9]\*[1-9][0-9]\*$　　//匹配正整数

^((-\d+)|(0+))$　　//匹配非正整数（负整数 + 0）

^-[0-9]\*[1-9][0-9]\*$　　//匹配负整数

^-?\d+$　　　　//匹配整数

^\d+(\.\d+)?$　　//匹配非负浮点数（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]\*))$　//匹配正浮点数

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

^(-?\d+)(\.\d+)?$