JavaScript 验证表单大全

```
<script>

/*
用途：校验 ip 地址的格式
输入：strIP：ip 地址
返回：如果通过验证返回 true,否则返回 false；

*/
function isIP(strIP) {
if (isNull(strIP)) return false;
var re=/^(\d+)\.(\d+)\.(\d+)\.(\d+)$/g //匹配 IP 地址的正则表达式
if(re.test(strIP))
{
if( RegExp.$1 <256 && RegExp.$2<256 && RegExp.$3<256 && RegExp.$4<256)
return true;
}
return false;
}

/*
用途：检查输入字符串是否为空或者全部都是空格
输入：str
返回：
如果全是空返回 true,否则返回 false
*/
function isNull( str ){
if ( str == "" ) return true;
var regu = "^[ ]+$";
var re = new RegExp(regu);
return re.test(str);
}
```

```
/*
用途：检查输入对象的值是否符合整数格式
输入：str  输入的字符串
返回：如果通过验证返回 true,否则返回 false

*/
function isInteger( str ){
var regu = /^[-]{0,1}[0-9]{1,}$/;
return regu.test(str);
}

/*
用途：检查输入手机号码是否正确
输入：
s：字符串
返回：
如果通过验证返回 true,否则返回 false

*/
function checkMobile( s ){
var regu =/^[1][3][0-9]{9}$/;
var re = new RegExp(regu);
if (re.test(s)) {
return true;
}else{
return false;
}
}


/*
用途：检查输入字符串是否符合正整数格式
输入：
s：字符串
返回：
```

```
如果通过验证返回 true,否则返回 false

*/
function isNumber( s ){
var regu = "^[0-9]+$";
var re = new RegExp(regu);
if (s.search(re) != -1) {
return true;
} else {
return false;
}
}

/*
用途：检查输入字符串是否是带小数的数字格式,可以是负数
输入：
s：字符串
返回：
如果通过验证返回 true,否则返回 false

*/
function isDecimal( str ){
if(isInteger(str)) return true;
var re = /^[-]{0,1}(\d+)[\.]+(\d+)$/;
if (re.test(str)) {
if(RegExp.$1==0&&RegExp.$2==0) return false;
return true;
} else {
return false;
}
}

/*
用途：检查输入对象的值是否符合端口号格式
输入：str 输入的字符串
返回：如果通过验证返回 true,否则返回 false
```

```
*/
function isPort( str ){
return (isNumber(str) && str<65536);
}

/*
用途：检查输入对象的值是否符合 E-Mail 格式
输入：str  输入的字符串
返回：如果通过验证返回 true,否则返回 false

*/
function isEmail( str ){
var myReg = /^[-_A-Za-z0-9]+@([_A-Za-z0-9]+\.)+[A-Za-z0-9]{2,3}$/;
if(myReg.test(str)) return true;
return false;
}

/*
用途：检查输入字符串是否符合金额格式
格式定义为带小数的正数，小数点后最多三位
输入：
s：字符串
返回：
如果通过验证返回 true,否则返回 false

*/
function isMoney( s ){
var regu = "^[0-9]+[\.][0-9]{0,3}$";
var re = new RegExp(regu);
if (re.test(s)) {
return true;
} else {
return false;
}
}
```

```
/*
用途：检查输入字符串是否只由英文字母和数字和下划线组成
输入：
s：字符串
返回：
如果通过验证返回 true,否则返回 false

*/
function isNumberOr_Letter( s ){//判断是否是数字或字母

var regu = "^[0-9a-zA-Z\_]+$";
var re = new RegExp(regu);
if (re.test(s)) {
return true;
}else{
return false;
}
}
/*
用途：检查输入字符串是否只由英文字母和数字组成
输入：
s：字符串
返回：
如果通过验证返回 true,否则返回 false

*/
function isNumberOrLetter( s ){//判断是否是数字或字母

var regu = "^[0-9a-zA-Z]+$";
var re = new RegExp(regu);
if (re.test(s)) {
return true;
}else{
return false;
}
}
```

```
/*
用途：检查输入字符串是否只由汉字、字母、数字组成
输入：
value：字符串
返回：
如果通过验证返回 true,否则返回 false

*/
function isChinaOrNumbOrLett( s ){//判断是否是汉字、字母、数字组成

var regu = "^[0-9a-zA-Z\u4e00-\u9fa5]+$";
var re = new RegExp(regu);
if (re.test(s)) {
return true;
}else{
return false;
}
}

/*
用途：判断是否是日期
输入：date：日期；fmt：日期格式
返回：如果通过验证返回 true,否则返回 false
*/
function isDate( date, fmt ) {
if (fmt==null) fmt="yyyyMMdd";
var yIndex = fmt.indexOf("yyyy");
if(yIndex==-1) return false;
var year = date.substring(yIndex,yIndex+4);
var mIndex = fmt.indexOf("MM");
if(mIndex==-1) return false;
var month = date.substring(mIndex,mIndex+2);
var dIndex = fmt.indexOf("dd");
if(dIndex==-1) return false;
var day = date.substring(dIndex,dIndex+2);
if(!isNumber(year)||year>"2100" || year< "1900") return false;
```

```
if(!isNumber(month)||month>"12" || month< "01") return false;
if(day>getMaxDay(year,month) || day< "01") return false;
return true;
}

function getMaxDay(year,month) {
if(month==4||month==6||month==9||month==11)
return "30";
if(month==2)
if(year%4==0&&year%100!=0 || year%400==0)
return "29";
else
return "28";
return "31";
}

/*
用途：字符1是否以字符串2结束
输入：str1：字符串；str2：被包含的字符串
返回：如果通过验证返回 true,否则返回 false

*/
function isLastMatch(str1,str2)
{
var index = str1.lastIndexOf(str2);
if(str1.length==index+str2.length) return true;
return false;
}



/*
用途：字符1是否以字符串2开始
输入：str1：字符串；str2：被包含的字符串
返回：如果通过验证返回 true,否则返回 false

*/
```

```
function isFirstMatch(str1,str2)
{
var index = str1.indexOf(str2);
if(index==0) return true;
return false;
}

/*
用途：字符1是包含字符串2
输入：str1：字符串；str2：被包含的字符串
返回：如果通过验证返回 true,否则返回 false

*/
function isMatch(str1,str2)
{
var index = str1.indexOf(str2);
if(index==-1) return false;
return true;
}


/*
用途：检查输入的起止日期是否正确，规则为两个日期的格式正确，
且结束如期>=起始日期
输入：
startDate：起始日期，字符串
endDate：结束如期，字符串
返回：
如果通过验证返回 true,否则返回 false

*/
function checkTwoDate( startDate,endDate ) {
if( !isDate(startDate) ) {
alert("起始日期不正确!");
return false;
} else if( !isDate(endDate) ) {
```

```
alert("终止日期不正确!");
return false;
} else if( startDate > endDate ) {
alert("起始日期不能大于终止日期!");
return false;
}
return true;
}

/*
用途：检查输入的 Email 信箱格式是否正确
输入：
strEmail：字符串
返回：
如果通过验证返回 true,否则返回 false

*/
function checkEmail(strEmail) {
//var emailReg = /^[_a-z0-9]+@([_a-z0-9]+\.)+[a-z0-9]{2,3}$/;
var emailReg = /^[\w-]+(\.[\w-]+)*@[\w-]+(\.[\w-]+)+$/;
if( emailReg.test(strEmail) ){
return true;
}else{
alert("您输入的 Email 地址格式不正确！");
return false;
}
}

/*
用途：检查输入的电话号码格式是否正确
输入：
strPhone：字符串
返回：
如果通过验证返回 true,否则返回 false

*/
```

```
function checkPhone( strPhone ) {
var phoneRegWithArea = /^[0][1-9]{2,3}-[0-9]{5,10}$/;
var phoneRegNoArea = /^[1-9]{1}[0-9]{5,8}$/;
var prompt = "您输入的电话号码不正确!"
if( strPhone.length > 9 ) {
if( phoneRegWithArea.test(strPhone) ){
return true;
}else{
alert( prompt );
return false;
}
}else{
if( phoneRegNoArea.test( strPhone ) ){
return true;
}else{
alert( prompt );
return false;
}
}
}

</script>
<form name=a onsubmit="return checkPhone(document.a.b.value);">
<INPUT TYPE="text" NAME="b">
<input type="submit" name="Submit" value="check">
</form>
```

==========

JavaScript 表单验证

表单验证一直是网页设计者头痛的问题,表单验证类 Validator 就是为解决这个问题而写的，旨在使设计者从纷繁复杂的表单验证中解放出来，把精力集中于网页的设计和功能上的改进上。

Validator 是基于 JavaScript 技术的伪静态类和对象的自定义属性，可以对网页中的表单项输入进行相应的验证，允许同一页面中同时验证多个表单，熟悉接口之后也可以对特定的表单项甚至仅仅是某个字符串进行验证。因为是伪静态类，所以在调用时不需要实例化，直接以"类名+.语法+属性或方法名"来调用。此外，Validator 还提供3种不同的错误提示模式，以满足不同的需要。

Validator 目前可实现的验证类型有：

1.是否为空；

2.中文字符；

3.双字节字符

4.英文；

5.数字；

6.整数；

7.实数；

8.Email 地址；

9.使用 HTTP 协议的网址；

10.电话号码；

11.货币；

12.手机号码；

13.邮政编码；

14.身份证号码(1.05增强)；

15.QQ 号码；

16.日期；

17.符合安全规则的密码；

18.某项的重复值；

19.两数的关系比较；

20.判断输入值是否在(n, m)区间；

21.输入字符长度限制(可按字节比较)；

22.对于具有相同名称的单选按钮的选中判断；

23.限制具有相同名称的多选按钮的选中数目；

24.自定义的正则表达式验证；

25.文件上传格式过滤(1.04新增)。

更新历史：

1.01

修正对12月份的日期验证(感谢 flylg999)

1.03

修正 Range 验证类型时将数字当字符串比较的 bug(感谢 cncom 和 xtlhnhbb)

修正日期验证(感谢 Papsam)

增加 Username 验证类型

增加对 Phone 验证类型时支持分机号

1.04

增加文件格式的过滤，用于上传时限制上传的文件格式

1.05
增强对身份证号码的验证
运行环境(客户端)：
在 Windows Server 2003下用 IE6.0+SP1和 Mozilla Firefox 1.0测试通过；
在 Lunix RedHat 9下的 Netscape 测试通过；
Validator 完整代码：
*/
<script>
/**
Validator v1.05
code by 我佛山人
[email]wfsr@msn.com[/email]
**/
Validator = {
Require : /.+/,
Email : /^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*$/,
Phone : /^((\(\d{2,3}\))|(\d{3}\-))?(\(0\d{2,3}\)|0\d{2,3}-)?[1-9]\d{6,7}(\-\d{1,4})?$/,
Mobile : /^((\(\d{2,3}\))|(\d{3}\-))?13\d{9}$/,
Url : /^http:\/\/[A-Za-z0-9]+\.[A-Za-z0-9]+[\/=\?%\-&_~`@[\]\':+!]*([^<>\"\"])*$/,
IdCard : "this.IsIdCard(value)",
Currency : /^\d+(\.\d+)?$/,
Number : /^\d+$/,
Zip : /^[1-9]\d{5}$/,
QQ : /^[1-9]\d{4,8}$/,
Integer : /^[-\+]?\d+$/,
Double : /^[-\+]?\d+(\.\d+)?$/,
English : /^[A-Za-z]+$/,
Chinese : /^[\u0391-\uFFE5]+$/,
Username : /^[a-z]\w{3,}$/i,
UnSafe : /^(([A-Z]*|[a-z]*|\d*|[-_\~!@#\$%\^&*\.\(\)\[\]\{\}<>\?\\\/\'\"]*)|.{0,5})$|\s/,
IsSafe : function(str){return !this.UnSafe.test(str);},
SafeString : "this.IsSafe(value)",
Filter : "this.DoFilter(value, getAttribute('accept'))",
Limit : "this.limit(value.length,getAttribute('min'), getAttribute('max'))",
LimitB : "this.limit(this.LenB(value), getAttribute('min'), getAttribute('max'))",
Date : "this.IsDate(value, getAttribute('min'), getAttribute('format'))",

```
Repeat : "value == document.getElementsByName(getAttribute('to'))[0].value",
Range : "getAttribute('min') < (value|0) && (value|0) < getAttribute('max')",
Compare : "this.compare(value,getAttribute('operator'),getAttribute('to'))",
Custom : "this.Exec(value, getAttribute('regexp'))",
Group : "this.MustChecked(getAttribute('name'), getAttribute('min'), getAttribute('max'))",
ErrorItem : [document.forms[0]],
ErrorMessage : ["以下原因导致提交失败：\t\t\t\t"],
Validate : function(theForm, mode){
var obj = theForm || event.srcElement;
var count = obj.elements.length;
this.ErrorMessage.length = 1;
this.ErrorItem.length = 1;
this.ErrorItem[0] = obj;
for(var i=0;i<count;i++){
with(obj.elements[i]){
var _dataType = getAttribute("dataType");
if(typeof(_dataType) == "object" || typeof(this[_dataType]) == "undefined") continue;
this.ClearState(obj.elements[i]);
if(getAttribute("require") == "false" && value == "") continue;
switch(_dataType){
case "IdCard" :
case "Date" :
case "Repeat" :
case "Range" :
case "Compare" :
case "Custom" :
case "Group" :
case "Limit" :
case "LimitB" :
case "SafeString" :
case "Filter" :
if(!eval(this[_dataType])) {
this.AddError(i, getAttribute("msg"));
}
break;
```

```
default :
if(!this[_dataType].test(value)){
this.AddError(i, getAttribute("msg"));
}
break;
}
}
}
if(this.ErrorMessage.length > 1){
mode = mode || 1;
var errCount = this.ErrorItem.length;
switch(mode){
case 2 :
for(var i=1;i<errCount;i++)
this.ErrorItem[i].style.color = "red";
case 1 :
alert(this.ErrorMessage.join("\n"));
this.ErrorItem[1].focus();
break;
case 3 :
for(var i=1;i<errCount;i++){
try{
var span = document.createElement("SPAN");
span.id = "__ErrorMessagePanel";
span.style.color = "red";
this.ErrorItem[i].parentNode.appendChild(span);
span.innerHTML = this.ErrorMessage[i].replace(/\d+:/,"*");
}
catch(e){alert(e.description);}
}
this.ErrorItem[1].focus();
break;
default :
alert(this.ErrorMessage.join("\n"));
break;
}
```

```
return false;
}
return true;
},
limit : function(len,min, max){
min = min || 0;
max = max || Number.MAX_VALUE;
return min <= len && len <= max;
},
LenB : function(str){
return str.replace(/[^\x00-\xff]/g,"**").length;
},
ClearState : function(elem){
with(elem){
if(style.color == "red")
style.color = "";
var lastNode = parentNode.childNodes[parentNode.childNodes.length-1];
if(lastNode.id == "__ErrorMessagePanel")
parentNode.removeChild(lastNode);
}
},
AddError : function(index, str){
this.ErrorItem[this.ErrorItem.length] = this.ErrorItem[0].elements[index];
this.ErrorMessage[this.ErrorMessage.length] = this.ErrorMessage.length + ":" + str;
},
Exec : function(op, reg){
return new RegExp(reg,"g").test(op);
},
compare : function(op1,operator,op2){
switch (operator) {
case "NotEqual":
return (op1 != op2);
case "GreaterThan":
return (op1 > op2);
case "GreaterThanEqual":
return (op1 >= op2);
```

```
case "LessThan":
return (op1 < op2);
case "LessThanEqual":
return (op1 <= op2);
default:
return (op1 == op2);
}
},
MustChecked : function(name, min, max){
var groups = document.getElementsByName(name);
var hasChecked = 0;
min = min || 1;
max = max || groups.length;
for(var i=groups.length-1;i>=0;i--)
if(groups[i].checked) hasChecked++;
return min <= hasChecked && hasChecked <= max;
},
DoFilter : function(input, filter){
return new RegExp("^.+\.(?=EXT)(EXT)$".replace(/EXT/g, filter.split(/\s*,\s*/).join("|")), "gi").test(input);
},
IsIdCard : function(number){
var date, Ai;
var verify = "10x98765432";
var Wi = [7, 9, 10, 5, 8, 4, 2, 1, 6, 3, 7, 9, 10, 5, 8, 4, 2];
var area = ['','','','','','','','','','','北京','天津','河北','山西','内蒙古','','','','','','辽宁','吉林','黑龙江','','','','','','','','上海','江苏','浙江','安微','福建','江西','山东','','','','河南','湖北','湖南','广东','广西','海南','','','','重庆','四川','贵州','云南','西藏','','','','','','','陕西','甘肃','青海','宁夏','新疆','','','','','','台湾','','','','','','','','','','香港','澳门','','','','','','','','','国外'];
var re = number.match(/^(\d{2})\d{4}(((\d{2})(\d{2})(\d{2})(\d{3}))|((\d{4})(\d{2})(\d{2})(\d{3}[x\d])))$/i);
if(re == null) return false;
if(re[1] >= area.length || area[re[1]] == "") return false;
if(re[2].length == 12){
Ai = number.substr(0, 17);
```

```
date = [re[9], re[10], re[11]].join("-");
}
else{
Ai = number.substr(0, 6) + "19" + number.substr(6);
date = ["19" + re[4], re[5], re[6]].join("-");
}
if(!this.IsDate(date, "ymd")) return false;
var sum = 0;
for(var i = 0;i<=16;i++){
sum += Ai.charAt(i) * Wi[i];
}
Ai += verify.charAt(sum%11);
return (number.length ==15 || number.length == 18 && number == Ai);
},
IsDate : function(op, formatString){
formatString = formatString || "ymd";
var m, year, month, day;
switch(formatString){
case "ymd" :
m = op.match(new RegExp("^((\\d{4})|(\\d{2}))([-./])(\\d{1,2})\\4(\\d{1,2})$"));
if(m == null ) return false;
day = m[6];
month = m[5]*1;
year = (m[2].length == 4) ? m[2] : GetFullYear(parseInt(m[3], 10));
break;
case "dmy" :
m = op.match(new RegExp("^(\\d{1,2})([-./])(\\d{1,2})\\2((\\d{4})|(\\d{2}))$"));
if(m == null ) return false;
day = m[1];
month = m[3]*1;
year = (m[5].length == 4) ? m[5] : GetFullYear(parseInt(m[6], 10));
break;
default :
break;
}
if(!parseInt(month)) return false;
```

month = month==0 ?12:month;
var date = new Date(year, month-1, day);
return (typeof(date) == "object" && year == date.getFullYear() && month == (date.getMonth()+1) && day == date.getDate());
function GetFullYear(y){return ((y<30 ? "20" : "19") + y)|0;}
}
}
</script>
/*
使用方法:
验证表单
在表单中加上 onsubmit 事件，触发调用 Validaotor 的 Validate 方法，代码示例:
CODE:
<form onSubmit="return Validator.Validate(this,3)" action="your_application_page" method="post">
... ...
</form>
//---------------------
Validate 方法有两个可选参数，第一个为表单对象，如果是写在表单的 onsubmit 事件中，可以用 this 指代当前表单，默认值为事件源对象；第二个参数为错误提示模式，可选值为1,2和3，默认值为1。省略第二个参数时相当于使用 Validate(objForm,1)，省略第一个参数时相当于 Validate(this,1)。注意，不可以省略第一个参数而只写第二个参数，Validate(,2)是错误的用法。
//////////////具体的实例
1. 验证输入是否 Email 地址
<input name="Email" dataType="Email" msg="信箱格式不正确">
2. 如果对 Email 地址有特殊的限制的话
<input name="Email" dataType="Custom" regexp="^\w+([-+.]\w+)*@\w+([-.]\\w+)*\.\w+([-.]\w+)*$" msg="信箱格式不正确">
3. Range 这个范围的使用方法
<input name="Range" dataType="Range" msg="超出了范围18 - 28 " min="18" max="28">

[说明]Validator 的必要属性是 dataType 和 msg(区分大小写)，然后根据 dataType 值的不同，会引发出不同的属性。因为程序中已经集成 Email 地址格式的正则，所以可以直接用 dateType="Email"进行验证，如果对 Email 地址的格式有不同的限制，可以用自定义的正则

来验证(参考第二段代码)。