



产品：Validator
编写：我佛山人
版本：1.0

表单的验证一直是网页设计者头痛的问题，表单验证类 Validator就是为了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性上的改进上。

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；
- 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；

运行环境(客户端)：


```
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 "Date" :
case "Repeat" :
case "Range" :
case "Compare" :
case "Custom" :
case "Group" :
case "Limit" :
case "LimitB" :
case "SafeString" :
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;
},
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})$"));
if(m == null ) return false;
day = m[6];
month = m[5]--;
year = (m[2].length == 4) ? m[2] : GetFullYear(parseInt(m[3], 10));
break;
case "dmy" :
m = op.match(new RegExp("^((\\d{1,2})([-./])(\\d{1,2})\\2((\\d{4})|
(\\d{2}))$"));
if(m == null ) return false;
day = m[1];
month = m[3]--;
year = (m[5].length == 4) ? m[5] : GetFullYear(parseInt(m[6], 10));
break;
default :
break;
}
if(!parseInt(month)) return false;
month = month==12 ?0:month;

```

```
var date = new Date(year, month, day);
return (typeof(date) == "object" && year == date.getFullYear() && month ==
date.getMonth() && day == date.getDate());
function GetFullYear(y){return ((y<30 ? "20" : "19") + y)|0;}
}
}
</script>
```

语法：`dataType="Require | Chinese | English | Number | Integer | Double | Email | Url | Phone | Mobile | Currency | Zip | IdCard | QQ | Date | SafeString | Repeat | Compare | Range | Limit | LimitB | Group | Custom"`

类型：字符串。必选。

说明：用于设定表单项的输入数据验证类型。

选值说明：

可选值	验证功能
Require	必填项
Chinese	中文
English	英文
Number	数字
Integer	整数
Double	实数
Email	Email地址格式
Url	基于HTTP协议的网址格式
Phone	电话号码格式
Mobile	手机号码格式
Currency	货币格式
Zip	邮政编码
IdCard	身份证号码
QQ	QQ号码
Date	日期
SafeString	安全密码
Repeat	重复输入
Compare	关系比较

Range	输入范围
Limit	限制输入长度
LimitB	限制输入的字节长度
Group	验证单/多选按钮组
Custom	自定义正则表达式验证

语法：`max="int"`

类型：字符串。在`dataType`属性值为`Range`时必选，为`Group`且待验证项是多选按钮组时可选(此时默认值为1)，为`Limit/LimitB`时可选(此时默认值为`1.7976931348623157e+308`，即`Number.MAX_VALUE`的值)。

说明：当`dataType`属性值为`Range`时，用于判断输入是否在`min`与`max`的属性值间；当`dataType`属性值为`Group`，且待验证项是多选按钮组时，用于设定多选按钮组的选中个数，判断选中个数是否在`[min, max]`区间；当`dataType`属性值为`Limit`时，用于验证输入的字符数是否在`[min, max]`区间；当`dataType`属性值为`LimitB`时，用于验证输入字符的字节数是否在`[min, max]`区间。

语法：`min="int"`

类型：字符串。在`dataType`属性值为`Range`时必选，为`Group`且待验证项是多选按钮组时可选(此时默认值为1)，为`Limit/LimitB`时可选(此时默认值为0)。

说明：当`dataType`属性值为`Range`时，用于判断输入是否在`min`与`max`的属性值间；当`dataType`属性值为`Group`，且待验证项是多选按钮组时，用于设定多选按钮组的选中个数，判断选中个数是否在`[min, max]`区间；当`dataType`属性值为`Limit`时，用于验证输入的字符数是否在`[min, max]`区间；当`dataType`属性值为`LimitB`时，用于验证输入字符的字节数是否在`[min, max]`区间。

语法：`msg="string"`

类型：字符串。必选。

说明：在验证失败时要提示的出错信息。

语法：**operator**="NotEqual | GreaterThan | GreaterThanEqual | LessThan | LessThanEqual | Equal"

类型：字符串。在**dataType**属性值为**Compare**时可选(默认值为**Equal**)。

说明：参考**to**属性。

各选值所对应的关系操作符：

可选值	意义说明
NotEqual	不等于 !=
GreaterThan	大于 >
GreaterThanEqual	大于等于 >=
LessThan	小于 <
LessThanEqual	小于等于 <=
Equal	等于 =

语法：`require="true | false"`

类型：字符串。可选。

说明：用于设定表单项的验证方式。当值为`false`时表单项不是必填项，但当有填写时，仍然要执行`dataType`属性所设定的验证方法，值为`true`或任何非`false`字符时可省略此属性。

语法：`to="string | int"`

类型：字符串。当**dataType**值为**Repeat**或**Compare**时必选。

说明：当**dataType**值为**Repeat**时，**to**的值为某表单项的**name**属性值，用于设定当前表单项的值是否与目标表单项的值一致；当**dataType**的值为**Compare**时，**to**的选值类型为实数，用于判断当前表单项的输入与**to**的值是否符合**operator**属性值所指定的关系。

语法：`format="ymd | dmy"`

类型：字符串。在`dataType`属性值为`Date`时可选(默认值为`ymd`)。

说明：用于验证输入是否为符合`format`属性值所指定格式的日期。

符合规则的输入示例：`2004-11-23`，`2004/11/23`，`04.11.23`，`23-11-2004`等

注意：当输入的年份为2位时，如果数值小于30，将使年份看作处于21世纪，否则为20世纪。

语法：`regexp="object"`

类型：字符串。在`dataType`属性值为Custom时必选。

说明：用于验证输入是否符合`regexp`属性所指定的正则表达式。

为便于理解，这里参考C++的语法列出Volidator的方法原型。

提示：

以Exec方法为例作说明。

```
static public bool Exec(string op, object reg)
```

<code>static</code>	表示静态方法，不需要实例化类即可用.语法使用。
---------------------	-------------------------

<code>public</code>	表示公开方法，可用.语法使用。
---------------------	-----------------

<code>bool</code>	表示方法返回的类型为布尔值true或false。
-------------------	--------------------------

<code>string</code> <code>op</code>	表示参数op为字符串型。
--	--------------

<code>object</code> <code>reg</code>	表示参数reg为对象型，这里是正则表达式对象。
---	-------------------------

另外，如果参数object reg表示为object reg="^.*\$"，则表示参数reg可选，在不指定reg参数时方法将按reg值为^.*\$进行操作。如果bool改为void，则表示该方法无返回值。

```
static public bool IsSafe(string str)
```

说明：测试字符串str是否符合安全规则----包含字母、数字和特殊符号的一种以上，不允许出现空间，至少需要6位的长度。

应用：dataType属性值为SafeString时的验证。

为便于理解，这里参考C++的语法列出Validator的方法原型。

提示：

以Exec方法为例作说明。

```
static public bool Exec(string op, object reg)
```

static	表示静态方法，不需要实例化类即可用.语法使用。
--------	-------------------------

public	表示公开方法，可用.语法使用。
--------	-----------------

bool	表示方法返回的类型为布尔值true或false。
------	--------------------------

string op	表示参数op为字符串型。
--------------	--------------

object reg	表示参数reg为对象型，这里是正则表达式对象。
---------------	-------------------------

另外，如果参数object reg表示为object reg="^.*\$"，则表示参数reg可选，在不指定reg参数时方法将按reg值为^.*\$进行操作。如果bool改为void，则表示该方法无返回值。

```
static public bool Validate(object theForm=event.srcElement, int mode=1)
```

说明：Validator的主方法，测试表单theForm是否符合验证要求，符合则返回true，否则以mode所指定的出错提示模式来提示错误。

应用：表单验证入口方法。

为便于理解，这里参考C++的语法列出Volidator的方法原型。

提示：

以Exec方法为例作说明。	
<code>static public bool Exec(string op, object reg)</code>	
<code>static</code>	表示静态方法，不需要实例化类即可用.语法使用。
<code>public</code>	表示公开方法，可用.语法使用。
<code>bool</code>	表示方法返回的类型为布尔值true或false。
<code>string</code> <code>op</code>	表示参数op为字符串型。
<code>object</code> <code>reg</code>	表示参数reg为对象型，这里是正则表达式对象。
另外，如果参数 <code>object reg</code> 表示为 <code>object reg="^.*\$"</code> ，则表示参数reg可选，在不指定reg参数时方法将按reg值为 <code>^.*\$</code> 进行操作。如果 <code>bool</code> 改为 <code>void</code> ，则表示该方法无返回值。	

`static public bool limit(int len, int min=0, int max)`

说明：测试输入字符的长度值len(字符数或字节数)，是否在区间[**min**, **max**]。

应用：`dataType`属性值为**Limit/LimitB**时的验证。

为便于理解，这里参考C++的语法列出Volidator的方法原型。

提示：

以Exec方法为例作说明。

```
static public bool Exec(string op, object reg)
```

static	表示静态方法，不需要实例化类即可用.语法使用。
--------	-------------------------

public	表示公开方法，可用.语法使用。
--------	-----------------

bool	表示方法返回的类型为布尔值true或false。
------	--------------------------

string op	表示参数op为字符串型。
--------------	--------------

object reg	表示参数reg为对象型，这里是正则表达式对象。
---------------	-------------------------

另外，如果参数object reg表示为object reg="^.*\$"，则表示参数reg可选，在不指定reg参数时方法将按reg值为^.*\$进行操作。如果bool改为void，则表示该方法无返回值。

```
static public int LenB(string str)
```

说明：获取输入字符串的字节数。一个中文字为两个字节。

应用：dataType属性值为Limit/LimitB时的验证。

为便于理解，这里参考C++的语法列出Volidator的方法原型。

提示：

以Exec方法为例作说明。

```
static public bool Exec(string op, object reg)
```

static	表示静态方法，不需要实例化类即可用.语法使用。
--------	-------------------------

public	表示公开方法，可用.语法使用。
--------	-----------------

bool	表示方法返回的类型为布尔值true或false。
------	--------------------------

string op	表示参数op为字符串型。
--------------	--------------

object reg	表示参数reg为对象型，这里是正则表达式对象。
---------------	-------------------------

另外，如果参数object reg表示为object reg="^.*\$"，则表示参数reg可选，在不指定reg参数时方法将按reg值为^.*\$进行操作。如果bool改为void，则表示该方法无返回值。

```
static public void ClearState(object elem)
```

说明：清除指定表单项elem的错误提示信息。

应用：Validate方法

为便于理解，这里参考C++的语法列出Validator的方法原型。

提示：

以Exec方法为例作说明。	
<code>static public bool Exec(string op, object reg)</code>	
<code>static</code>	表示静态方法，不需要实例化类即可用.语法使用。
<code>public</code>	表示公开方法，可用.语法使用。
<code>bool</code>	表示方法返回的类型为布尔值true或false。
<code>string op</code>	表示参数op为字符串型。
<code>object reg</code>	表示参数reg为对象型，这里是正则表达式对象。
另外，如果参数object reg表示为object reg="^.*\$"，则表示参数reg可选，在不指定reg参数时方法将按reg值为^.*\$进行操作。如果bool改为void，则表示该方法无返回值。	

`static public void AddError(int index, string str)`

说明：将未通过验证的表单项的当前索引值index和错误提示信息str添加到Validator的ErrorMessage和ErrorItem数组。

应用：Validate方法

为便于理解，这里参考C++的语法列出Volidator的方法原型。

提示：

以Exec方法为例作说明。	
<code>static public bool Exec(string op, object reg)</code>	
<code>static</code>	表示静态方法，不需要实例化类即可用.语法使用。
<code>public</code>	表示公开方法，可用.语法使用。
<code>bool</code>	表示方法返回的类型为布尔值true或false。
<code>string op</code>	表示参数op为字符串型。
<code>object reg</code>	表示参数reg为对象型，这里是正则表达式对象。
另外，如果参数object reg表示为object reg="^.*\$"，则表示参数reg可选，在不指定reg参数时方法将按reg值为^.*\$进行操作。如果bool改为void，则表示该方法无返回值。	

`static public bool Exec(string op, object reg)`

说明：测试字符串op是否符合正则对象reg所设定的规则，是则返回true，否则返回false。

应用：当dateType属性值为Custom时的验证

为便于理解，这里参考C++的语法列出Validator的方法原型。

提示：

以Exec方法为例作说明。

```
static public bool Exec(string op, object reg)
```

static	表示静态方法，不需要实例化类即可用.语法使用。
--------	-------------------------

public	表示公开方法，可用.语法使用。
--------	-----------------

bool	表示方法返回的类型为布尔值true或false。
------	--------------------------

string op	表示参数op为字符串型。
--------------	--------------

object reg	表示参数reg为对象型，这里是正则表达式对象。
---------------	-------------------------

另外，如果参数object reg表示为object reg="^.*\$"，则表示参数reg可选，在不指定reg参数时方法将按reg值为^.*\$进行操作。如果bool改为void，则表示该方法无返回值。

```
static public bool compare(int op1, string operator, int op2)
```

说明：比较op1和op2是否符合operator所指定的关系，是则返回true,否则返回false。

应用：当dateType属性值为Compare时的验证

为便于理解，这里参考C++的语法列出Validator的方法原型。

提示：

以Exec方法为例作说明。	
<code>static public bool Exec(string op, object reg)</code>	
<code>static</code>	表示静态方法，不需要实例化类即可用.语法使用。
<code>public</code>	表示公开方法，可用.语法使用。
<code>bool</code>	表示方法返回的类型为布尔值true或false。
<code>string op</code>	表示参数op为字符串型。
<code>object reg</code>	表示参数reg为对象型，这里是正则表达式对象。
另外，如果参数object reg表示为object reg="^.*\$"，则表示参数reg可选，在不指定reg参数时方法将按reg值为^.*\$进行操作。如果bool改为void，则表示该方法无返回值。	

`static public bool MustChecked(string name, int min=1, int max=*)`

说明：测试名称为参数name所指定的单/多选按钮组的选中个数是否在[`min`, `max`]区间，是则返回true,否则返回false。

注意，对于单选按钮组，`min`和`max`属性没有意义，而对于多选按钮组，在不指定`max`值时，默认值为多选按钮组的多选按钮个数。

应用：当dateType属性值为Group时的验证

为便于理解，这里参考C++的语法列出Validator的方法原型。

提示：

以Exec方法为例作说明。

```
static public bool Exec(string op, object reg)
```

static	表示静态方法，不需要实例化类即可用.语法使用。
--------	-------------------------

public	表示公开方法，可用.语法使用。
--------	-----------------

bool	表示方法返回的类型为布尔值true或false。
------	--------------------------

string op	表示参数op为字符串型。
--------------	--------------

object reg	表示参数reg为对象型，这里是正则表达式对象。
---------------	-------------------------

另外，如果参数object reg表示为object reg="^.*\$"，则表示参数reg可选，在不指定reg参数时方法将按reg值为^.*\$进行操作。如果bool改为void，则表示该方法无返回值。

```
static public bool IsDate(string op, string formatString)
```

说明：测试op是否条例formatString所指定的日期格式，是则返回true,否则返回false。

应用：当dateType属性值为Date时的验证

验证表单

在表单中加上onsubmit事件，触发调用Validator的Validate方法，代码示例：

```
<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。

验证输入是否是Email地址

代码示例：

```
<input name="Email" dataType="Email" msg="信箱格式不正确">
```

或

```
<input name="Email" dataType="Custom" regexp="^\w+([-+.] \w+)*@\w+([-.] \w+)*\.\w+([-.] \w+)*$" msg="信箱格式不正确">
```

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)。

验证下拉菜单是否选中

代码示例：

```
<select name="Operation" dataType="Require" msg="未选择所用操作系统"
>
<option value="">选择您所用的操作系统</option>
<option value="Win98">Win98</option>
<option value="Win2k">Win2k</option>
<option value="WinXP">WinXP</option>
</select>
```

注意，对于IE，在option中没写value属性时IE的解释引擎将自动设置其值为空，而Firefox时将自动设置其值为text属性址。例如，在示例代码中如果第一个option不写value属性，IE中将得到value为空，而Firefox为"选择您所用的操作系统"。

验证是否选中单选按钮组中的一个

代码示例：

```
广东<input name="Province" value="1" type="radio">  
陕西<input name="Province" value="2" type="radio">  
浙江<input name="Province" value="3" type="radio">  
江西<input name="Province" value="4" type="radio" dataType="Group"  
msg="必须选定一个省份" >
```

对于单/多选按钮组的验证，**dataType**属性都为Group，然后只需在按钮组的最后一个写上**dataType**和**msg**属性。

注意，要成为单/多选按钮组，它们必须具有相同的name属性值。

限制多选按钮组的选中个数

代码示例：

```
运动<input name="Favorite" value="1" type="checkbox">  
上网<input name="Favorite" value="2" type="checkbox">  
听音乐<input name="Favorite" value="3" type="checkbox">  
看书<input name="Favorite" value="4" type="checkbox" data-type="Group"  
min="2" max="3" msg="必须选择2~3种爱好">
```

要限制多选按钮组的选中个数，必须设置min和max属性。min属性用于设定选中个数的下限，max为上限，默认时min为1，max为多选按钮组的个数。