

jQuery.validate.js API

(HappyCZX 整理)

名称	返回类型	描述
validate(options)	返回:Validator	验证所选的 FORM
valid()	返回:Boolean	检查是否验证通过
rules()	返回:Options	返回元素的验证规则
rules("add", rules)	返回:Options	增加验证规则
rules("remove", rules)	返回:Options	删除验证规则
removeAttrs(attributes)	返回:Options	删除特殊属性并且返回他们
Custom selectors		
:blank	返回:Validator	没有值的筛选器
:filled	返回:Array <Element >	有值的筛选器
:unchecked	返回:Array <Element >	没选择的元素的筛选器
Utilities		
jQuery.format (template, argument , argumentN...)	返回:String	用参数代替模板中的 {n}

Validator:

validate 方法返回一个 Validator 对象,它有很多方法,让你能使用引发校验程序或者改变 form 的内容.
validator 对象有很多方法,但下面只是列出常用的

form()	返回:Boolean	验证 form 返回成功还是失败
element(element)	返回:Boolean	验证单个元素是成功还是失败
resetForm()	返回:undefined	把前面验证的 FORM 恢复到验证前原来的状态
showErrors(errors)	返回:undefined	显示特定的错误信息

Validator functions:

setDefault(defaults)	返回:undefined	改变默认的设置
addMethod(name, method, message)	返回:undefined	添加一个新的验证方法. 必须包括一个独一无二的名字, 一个 JAVASCRIPT 的方法和一个默认的信息
addClassRules(name, rules)	返回:undefined	增加组合验证类型 在一个类里面用多种验证方法里比较有用
addClassRules(rules)	返回:undefined	增加组合验证类型 在一个类里面用多种验证方法里比较有用, 这个是一下子加多个

内置验证方式:

required()	返回: Boolean	必填验证元素
required(dependency-expression)	返回: Boolean	必填元素依赖于表达式的结果
required(dependency-callback)	返回: Boolean	必填元素依赖于回调函数的结果
remote(url)	返回: Boolean	请求远程校验。url 通常是一个远程调用方法
minlength(length)	返回: Boolean	设置最小长度
maxlength(length)	返回: Boolean	设置最大长度
rangelength(range)	返回: Boolean	设置一个长度范围[min, max]
min(value)	返回: Boolean	设置最大值
max(value)	返回: Boolean	设置最小值
email()	返回: Boolean	验证电子邮箱格式
range(range)	返回: Boolean	设置值的范围
url()	返回: Boolean	验证 URL 格式
date()	返回: Boolean	验证日期格式(类似 30/30/2008 的格式, 不验证日期准确性只验证格式)
dateISO()	返回: Boolean	验证 ISO 类型的日期格式
dateDE()	返回: Boolean	验证德式的日期格式 (29. 04. 1994 or 1. 1. 2006)
number()	返回: Boolean	验证十进制数字 (包括小数的)
digits()	返回: Boolean	验证整数
creditcard()	返回: Boolean	验证信用卡号
accept(extension)	返回: Boolean	验证相同后缀名的字符串
equalTo(other)	返回: Boolean	验证两个输入框的内容是否相同
phoneUS()	返回: Boolean	验证美式的电话号码

validate () 的可选项:

debug: 进行调试模式 (表单不提交):	<pre>\$(".selector").validate({ debug: true })</pre>
把调试设置为默认:	<pre>\$.validator.defaults({ debug: true })</pre>
submitHandler: 通过验证后运行的函数, 里面要加上表单提交的函数, 否则表单不会提交	<pre>\$(".selector").validate({ submitHandler: function(form) { \$(form).ajaxSubmit(); } })</pre>

<p>ignore: 对某些元素不进行验证</p>	<pre>\$("#myform").validate({ ignore:".ignore" })</pre>
<p>rules: 自定义规则, key:value 的形式, key 是要验证的元素, value 可以是字符串或对象</p>	<pre>\$(".selector").validate({ rules:{ name:"required", email:{ required:true, email:true } } })</pre>
<p>messages: 自定义的提示信息 key:value 的形式 key 是要验证的元素, 值是字符串或函数</p>	<pre>\$(".selector").validate({ rules:{ name:"required", email:{ required:true, email:true } }, messages:{ name:"Name 不能为空", email:{ required:"E-mail 不能为空", email:"E-mail 地址不正确" } } })</pre>
<p>groups: 对一组元素的验证, 用一个错误提示, 用 error Placement 控制把出错信息放在哪里</p>	<pre>\$("#myform").validate({ groups:{ username:"fname lname" }, errorPlacement:function(error, element) { if (element.attr("name") == "fname" element.attr("name") == "lname") error.insertAfter("#lastname"); else error.insertAfter(element); }, debug:true })</pre>
<p>Onsubmit Boolean 默认:true 是否提交时验证</p>	<pre>\$(".selector").validate({ onsubmit:false })</pre>

onfocusout Boolean 默认:true 是否在获取焦点时验证	<pre>\$(".selector").validate({ onfocusout:false })</pre>
onkeyup Boolean 默认:true 是否在敲击键盘时验证	<pre>\$(".selector").validate({ onkeyup:false })</pre>
onclick Boolean 默认:true 是否在鼠标点击时验证 (一般验证 checkbox, radiobox)	<pre>\$(".selector").validate({ onclick:false })</pre>
focusInvalid Boolean 默认:true 提交表单后, 未通过验证的表单(第一个或提交之前获得焦点的未通过验证的表单)会获得焦点	<pre>\$(".selector").validate({ focusInvalid:false })</pre>
focusCleanup Boolean 默认:false 当未通过验证的元素获得焦点时, 并移除错误提示 (避免和 focusInvalid. 一起使用)	<pre>\$(".selector").validate({ focusCleanup:true })</pre>
errorClass String 默认:"error" 指定错误提示的 css 类名, 可以自定义错误提示的样式	<pre>\$(".selector").validate({ errorClass:"invalid" })</pre>
errorElement String 默认:"label" 使用什么标签标记错误	<pre>\$(".selector").validate(errorElement:"em")</pre>
wrapper String 使用什么标签再把上边的 errorElement 包起来	<pre>\$(".selector").validate({ wrapper:"li" })</pre>
errorLabelContainer Selector 把错误信息统一放在一个容器里面	<pre>\$("#myform").validate({ errorLabelContainer:"#messageBox", wrapper:"li", submitHandler:function() { alert("Submitted!") } })</pre>
showErrors: 跟一个函数, 可以显示总共有多少个未通过验证的元素	<pre>\$(".selector").validate({ showErrors:function(errorMap, errorList) { \$("#summary").html("Your form contains " + this.numberOfInvalids() + " errors, see details below."); this.defaultShowErrors(); } })</pre>
errorPlacement: 跟一个函数, 可以自定义错误放到哪里	<pre>\$("#myform").validate({ errorPlacement:function(error, element) { error.appendTo(element.parent("td").next("td")) }; })</pre>

	<pre> }, debug:true })) </pre>
success: 要验证的元素通过验证后的动作, 如果跟一个字符串, 会当做一个 css 类, 也可跟一个函数	<pre> \$("#myform").validate({ success:"valid", submitHandler:function() { alert("Submitted!") } }) </pre>
highlight: 可以给未通过验证的元素加效果, 闪烁等	

addMethod(name, method, message) 方法:

参数 name 是添加的方法的名字

参数 method 是一个函数, 接收三个参数 (value, element, param) value 是元素的值, element 是元素本身 param 是参数, 我们可以用 addMethod 来添加除 built-in Validation methods 之外的验证方法 比如有一个字段, 只能输一个字母, 范围是 a-f, 写法如下:

```

$.validator.addMethod("af", function(value, element, params) {
    if(value.length>1) {
        return false;
    }
    if(value>=params[0] && value<=params[1]) {
        return true;
    }else{
        return false;
    }
}

```

}, "必须是一个字母, 且 a-f");

用的时候, 比如有个表单字段的 id="username", 则在 rules 中写

```

username: {
    af: ["a", "f"]
}

```

addMethod 的第一个参数, 就是添加的验证方法的名字, 这时是 af

addMethod 的第三个参数, 就是自定义的错误提示, 这里的提示为: "必须是一个字母, 且 a-f"

addMethod 的第二个参数, 是一个函数, 这个比较重要, 决定了用这个验证方法时的写法

如果只有一个参数, 直接写, 如果 af: "a", 那么 a 就是这个唯一的参数, 如果多个参数, 用在 [] 里, 用逗号分开

meta String 方式:

```

$("#myform").validate({
    meta:"validate",
    submitHandler:function() { alert("Submitted!") }
}

```

```
}}
```

```
<script type="text/javascript" src="js/jquery.metadata.js"></script>
```

```
<script type="text/javascript" src="js/jquery.validate.js"></script>
```

```
<form id="myform">
```

```
  <input type="text" name="email" class="{validate:{ required:true, email:true }}" />
```

```
  <input type="submit" value="Submit" />
```

```
</form>
```