jQuery插件validation使用

# 使用例子

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">  <meta charset="UTF-8">  <title>jQuery Validation 插件</title>  <link rel="stylesheet" href="style.css"/>  </head>  <body>  <form id="demoForm">  <fieldset>  <legend>用户登录</legend>  <p id="info"></p>  <p>  <label for="username">用户名</label>  <input type="text" id="username" name="username"/>  </p>  <p>  <label for="password">密码</label>  <input type="password" id="password" name="password"/>  </p>  <p>  <label for="confirm-password">确认密码</label>  <input type="password" id="confirm-password" name="confirm-password"/>  </p>  <p>  <input type="submit" value="登录"/>  </p>  </fieldset>  </form>  <script src="vendor/jquery-1.10.0.js"></script>  <script src="vendor/jquery.validate-1.13.1.js"></script>  <script>  var validator1;  $(document).ready(function () {  validator1 = $("#demoForm").validate({  debug: true,  rules: {  username: {  required: true,  minlength: 2,  maxlength: 10  },  password: {  required: true,  minlength: 2,  maxlength: 16  },  "confirm-password": {  equalTo: "#password"  }  },  messages: {  username: {  required: '请输入用户名',  minlength: '用户名不能小于2个字符',  maxlength: '用户名不能超过10个字符',  remote: '用户名不存在'  },  password: {  required: '请输入密码',  minlength: '密码不能小于2个字符',  maxlength: '密码不能超过16个字符'  },  "confirm-password": {  equalTo: "两次输入密码不一致"  }  },  highlight: function(element, errorClass, validClass) {  $(element).addClass(errorClass).removeClass(validClass);  $(element).fadeOut().fadeIn();  },  unhighlight: function(element, errorClass, validClass) {  $(element).removeClass(errorClass).addClass(validClass);  },  submitHandler: function (form) {  console.log($(form).serialize())  }  });  $("#check").click(function () {  console.log($("#demoForm").valid() ? "填写正确" : "填写不正确");  });  });  </script>  </body>  </html> |

# method:验证方法

validate()方法为核心方法，定义了基本的校验规则和一些有用的配置项。会返回一个**Validator**对象。

## 基本验证方法

**required**：必填项(不能为空项时使用，如用户名，密码)

**remote**：远程校验（比如需要看用户名是否存在,填写是否正确，进入后台数据库查询。）

**minlength**：字符最小长度

**maxlength**：字符最大长度

**rangelength**：长度范围（rangelength：[2,10]代表输入字符长度在两个到十个之间）

**min**：数值最小值

**max**：数值最大值

**range**：数值范围（range：[2,10]代表输入数字范围为2到10之间）

**email** ：Email格式（为true时，要求输入合法正确email格式才可以，@）

**url** ：URL格式（为true时，要求输入合法正确URL格式才可以，http：//之类的）

**date**：日期（多种格式日期都支持）

**dateISO**： ISO日期（只支持 四位年-月-日或 四位年/月/日 或-/交互使用）

**number**：数字（多种支持，负数，小数）

**digits**：整数（只支持 正整数）

**equalTo**：与另一个元素值相等（输入密码后的确认密码0

例如： "confirm-password":{equalTo:"#password"）}

## 高级验证方法

#1. **valid()方法**：检查表单或某些元素是否有效。

#2. **rules()方法**：

①无参数时：获取表单【元素】的校验规则（即：username可以，但demo  
Form不行）

②（"add",{min:2,max:10}）或（"remove","min max"）向表单元素添加或删除校验规则

#3.**Validator对象**

①validate()方法返回Validator对象，其有很多有用的方法，课程中是在控制台演示的。

|  |
| --- |
| **Validator.form()**： 验证表单是否有效，返回 true/false  **Validator.element("#username")** ：验证某个元素是否有效，返回true/false  **Validator.resetForm()** ：把表单恢复到验证前原来的状态(去除校验信息)  **Validator.showErrors({username:"你填错了",password:" "})** ：针对某个元素显示特定的错误信息  **Validator.numberOfInvalids()**： 返回无效的元素数量 |

②Validator对象还有很多有用的静态方法，可直接使用，不用var validator后再validator=....

|  |
| --- |
| **$.validator.addMethod(name,method [,message] )**：增加自定义的验证方法  **$.validator.format(template,argument,argumentN...)**：格式化字符串，用参数代替模板中的{n}  例如：var template=$.vaildator.format("{0}-{1}-{2}")；template("值在",["2","10"]，“之间”)  **$.validator.setDefaults(options)**：修改插件默认设置  **$.validator.addClassRules(name,rules)**：为某些包含名为name的class增加组合验证类型。 |

#4. **validate()方法配置项**

**debug**:true：用于 调试，表单不会提交

**submitHandler**：通过【验证后】运行的函数，可以加上表单提交的方法

|  |
| --- |
| submitHandler:function(from){  from:submit();  $(form).Ajax.submit();//$ajax等方式提交表单  }, |

**invalidHandler**：【无效】表单提交后运行的函数

|  |
| --- |
| invalidHandler:function(event,validator){  ...  }  也可以写一个事件来触发$("选择器属性值").on("事件名",function(event,validator)){}); |

**ignore：**("默认:hidden") 对某些元素不进行验证（默认不对隐藏的元素进行验证）

**rules**： 定义校验规则（每条规则都有depends属性，true才执行。）

|  |
| --- |
| minlength:{  param:2,  depends：function（element）{  return $("#password").is(":filled");  }  } |

**messages** ：定义提示信息

**groups** ：对一组元素的验证，用一个错误提示，用errorPlacement控制把出错信息放在那里（）

例如： groups:{login:"username password confirm-password"} //对一组元素进行验证

**errorPlacement**:function(error,element){ //对一组元素验证后错误信息显示error.insertBefore("#info");//显示在id为info的位置上}

**onsubmit**：是否在提交时验证

**onfocusout**：是否在获取焦点时验证

**onkeyup**：是否在敲击键盘时验证

**onclick**：是否在鼠标点击时验证，一般用于checkbox或者radio

**focusInvalid**：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）是否会获得焦点

**focusCleanup**：当未通过验证的元素获得焦点时，是否移除错误提示

**errorClass**指定错误提示的css类名，默认为error,可以自定义

**validClass**指定验证通过的css类名

**errorElement：**使用什么标签标记错误 （如<li>标记）

**wrapper**：使用什么标签把上边的errorElement包起来 （如<ul>包裹）

**errorLabelContainer**：把错误信息统一放在一个容器里面 （如放在类或id选择器中）

**errorContainer**：显示或者隐藏验证信息，可以自动实现由错误信息出现时把容器属性变为显示，无错误时隐藏（以上四个一般同时使用）

**showError**：可以显示总共有多少个未通过验证的元素

**errorPlacement**：自定义错误信息放到哪里

|  |
| --- |
| errorPlacement: **function** (error, element) {  **if** ($(element).is('select')) {  error.appendTo($(element).parent());  }  }  //error表示提示信息代码。element表示当前具体表单项对象 |

**success**：要验证的元素通过验证后的动作（主要针对的是l验证标签元素）

**highlight**：可以给未通过验证的元素加效果（主要针对input元素）

**unhighlight**：去除未通过验证的元素的效果，一般和highlight同时使用

#5. **选择器扩展**:

|  |
| --- |
| :blank 选择所有值为空的元素  :filed 选择所有值不为空的元素  :unchecked 选择所有没有被选中的元素（一般针对复选框） |

# validation常见使用

## 自定义验证方法

|  |
| --- |
| aaddMethod(name,method [,message])  -- name：方法名称  -- method：function(value,element,params)方法逻辑  --value=验证元素的值  --element=验证元素  --params 可选参数  message：提示信息 |
| $.validator.addMethod("postcode", function(value, element,params) {  var postcode= /^[0-9]{6}$/;  return this.optional(element) || (postcode.test(value));}  , $.validator.format("请填写正确的{0}邮编!"));  说明：this.optional(element);当邮政编码不是必填项时，element为空，返回true，不会进行效验  例如： username:{required:true,postcode:true //或 postcode:"中国" -----这里的"中国"会被传给{0}} |
| // 手机号码验证  jQuery.validator.addMethod("isMobile", function(value, element) {  var length = value.length;  var mobile = /^(13[0-9]{9})|(18[0-9]{9})|(14[0-9]{9})|(17[0-9]{9})|(15[0-9]{9})$/;  return this.optional(element) || (length == 11 && mobile.test(value));  }, "请正确填写您的手机号码"); |
| $("#checkPhoneForm").validate({  rules : {  phone : {  required : true,  minlength : 11,  // 自定义方法：校验手机号在数据库中是否存在  // checkPhoneExist : true,  **isMobile : true**  }  },  messages : {  phone : {  required : "请输入手机号",  minlength : "确认手机不能小于11个字符",  **isMobile : "请正确填写您的手机号码"**  }  }  }); |

## 远程校验

|  |
| --- |
| #html  type: {  remote:{  url: "changeusertype.json",  type: "get",  dataType: 'json',  data: {  'type': function(){return $('input[name="type"]:checked').val();}  }  }  注意：data没有的时候，默认会把type传到服务器端。 |