### jquery.validate.pack.js

### 介绍：

jquery.validate.js是jquery旗下的一个验证框架，借助jquery的优势，我们可以迅速验证一些常见的输入,并且可以自己扩充自己的验证方法。

jQuery.Validate是监控form的，在任何提交表单的操作前jQuery.Validate都会检测表单里的输入项是否满足规则，满足才允许提交。所以需要在jQuery(document).ready（）时为form进行验证注册，格式： $(“#formid”).validate(jsonobj)。

其中formid是文件中form表单的id，jsonobj包含两个属性：rules和messages，rules用来指明每个字段上添加的校验规则，messages用来说明相应的校验出错时对应的提示语。

messages为非必须，如果自己不定义该属性，会采用默认的提示。

### 项目引用：

|  |
| --- |
| <html xmlns="http://www.w3.org/1999/xhtml">  <head>  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />  <title>表单校验</title>  <script type="text/javascript" src=../../"/common/jquery.js"></script>  <script type="text/javascript" src=../../"/common/jquery.validate.pack.js"></script>  <script type="text/javascript" src=../../"/common/jquery.form.js"></script>  </head>  <body>  <form id="test">  <input name="nameput" type="text" size="20"/>  <input name="submit" type="submit" value="提交"/>  </form>  <script type="text/javascript">  $("#test").validate({  rules:{  "nameput":{  required:true,  minlength:3,  maxlength:10  }  },  messages:{  "nameput":{  required:"不能为空",  minlength:jQuery.format("长度不小于{0}"),  maxlength:jQuery.format("长度不大于{0}")  }  }  })  </script>  </body>  </html>  例如/test/validate/formvalidate.jsp:  <html xmlns="http://www.w3.org/1999/xhtml">  <head>  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />  <title>表单校验</title>  <script type="text/javascript" src=../../"/common/jquery.js"></script>  <script type="text/javascript" src=../../"/common/jquery.validate.pack.js"></script>  <script type="text/javascript" src=../../"/common/jquery.form.js"></script>  </head>  <body>  <form id="test">  <input name="nameput" type="text" size="20"/>  <input name="submit" type="submit" value="提交"/>  </form>  <script type="text/javascript">  $("#test").validate({  rules:{  "nameput":{  required:true,  minlength:3,  maxlength:10  }  },  messages:{  "nameput":{  required:"不能为空",  minlength:jQuery.format("长度不小于{0}"),  maxlength:jQuery.format("长度不大于{0}")  }  }  })  </script>  </body>  </html>  注意：nameput是input标签的name。  Required、minlength、maxlength都是jQuery校验的内置验证方式。jQuery内置验证方式参考jQuery.validate.js的API。    3.自定义验证方式  下面是自定义的验证方式：  $.validator.addMethod("stringlength", function(value, element,params) {  //默认值 : {trim:true,minLength:-1,maxLength:-1  params = $.extend([true,-1,-1],params); //对于默认参数支持  if(params[0]){ //过滤首尾空格  value=$.trim(value);  }  value = value.replace(/<(?:.)\*?>/g,""); //验证时过滤标签  return this.optional(element) || ((params[1]<0 || value.length>=params[1])&&(params[2]<0 || value.length<=params[2]));  }, jQuery.format("长度在{1}-{2}之间"));  例如home工程中的登录校验：  $('#loginform').validate({//登陆校验  rules:{  "userAccount.userName":{  "requiredstring":["true"],  " requiredstring ":true,  "stringlength":["true","3","40"]  },  "userAccount.userPwd":{  "requiredstring":["true"],  "stringlength":["true","1","20"]  }  },  messages:{  "userAccount.userName":{  "requiredstring":"用户名必填",  "stringlength":jQuery.format("用户名长度在{1}和{2}之间")  },  "userAccount.userPwd":{  "requiredstring":"密码不可以为空",  "stringlength":jQuery.format("密码长度在{1}和{2}之间")  }  }  })  userAccount.userName是页面对应的input的name，requiredstring、requiredstring、stringlength是自己定义的校验，定义在/image/hi/common/js/zxwvalidate.js里。  ｛1｝、｛2｝等是rules里面对应验证方式的第几个元素，从0开始。  简单的实例：  $.validator.addMethod("twd", function(value, element,params) { //默认值 : {trim:true,minLength:-1,maxLength:-1  params = $.extend([true,-1,-1],params); //对于默认参数支持  if(params[0]){  value=$.trim(value);  }  })  $("#test").validate({  rules:{  "nameput":{  "twd":[true,3,10]  }  },  messages:{  "nameput":{  "twd":jQuery.format("长度在{1}和{2}之间")  }  }  })    4.其他注意事项  （1）校验默认是在点击提交submit的时候起作用.  （2）如果缺少$().ready(function() { })，校验内容必须写在表单的后面。  （3）debug方法需要单独写或者rules和messages的后面，否则不会起作用。    附：  jQuery.Validate为我们提供了3种验证编写方式，各有优缺点：  （1）在input对象中书写class样式指定验证规则或属性验证规则：  如<input type=”text” class=”required”/>  最简单、最便捷，提示消息使用jQuery.Validate的内置的消息（自定义扩展验证规则也属于此项），但是由于是以样式名的方式进行验证，导致了日后修改必须找到相应的input对象，同时无法使用高级验证规则。  （2）在input对象中书写class样式，只不过书写的方式改为了JSON格式，但是这种方式提供了自定义验证消息的支持：  如<input type=”text” class="{required:true,minlength:5,,messages:{required:'请输入内容'}”/>  简单、方便，但个人认为有点臃肿，还是和第1条一样和相对应的input对象纠缠在一起，并且还增加消息自定义，使得input对象变的更大了，干扰了页面代码的阅读，但可以使用高级验证规则（实际就是将第3种JS以JSON的格式放到具体的class中。  （3）使用纯JS的方式：  如：  $().ready(function() {  $("#aspnetform").validate({  rules: {  name: "required",  email: {  required: true,  email: true  }  })  })  很好的解决了HTML和验证规则的分离，就是书写较为麻烦，需要单独写JS脚本，但好处是可以统一验证规范，将每个页面的验证规则都写在头部的脚本中，方便日后维护。 |