附录 F

插件 API

F.1 Validation 插件 API

Validation 插件有两个经常被用到的选项，分别是方法（method）和规则（rule）。

（1）方法。验证方法就是通过执行验证逻辑判断一个元素是否合法。例如 email()方法就是检查当前文本格式是否是正确的 E-mail 格式。读者能很方便地利用 Validation 插件提供的方法来完成验证。另外，读者也可以自定义方法。

（2）规则。验证规则将元素和元素的验证方法关联起来，例如验证一个需要 E-mail 格式和必填的属性 name 为 email 的元素，可以定义该元素的规则如下：

```
email: {
    required: true,
    email:    true
}
```

- 插件方法如表 F-1 所示。

表 F-1 插件方法

名　称	返回类型	说　明
validate(options)	Validator	验证被选择的 form
valid( )	Boolean	检查被选择的 from 或者被选择的所有元素是否有效
rules( )	Options	为第 1 个被选择的元素返回验证规则
rules("add", rules)	Options	增加指定的验证规则并为第 1 个匹配元素返回所有的规则
rules("remove", rules)	Options	移除指定的验证规则并为第 1 个匹配元素返回所有的规则
removeAttrs(attributes)	Options	从第 1 个匹配元素中移除指定的属性并返回

- 内置验证规则如表 F-2 所示。

表 F-2　　内置验证规则

名　称	返回类型	说　明
required( )	Boolean	使元素总是必须的
required(dependency-expression)	Boolean	根据给定的表达式结果，判断元素是否是必须的
required(dependency-callback)	Boolean	根据给定的回调函数的返回值，判断元素是否是必须的
remote(url)	Boolean	使用请求资源检查元素的有效性
minlength(length)	Boolean	要求元素满足给定的最小长度规则
maxlength(length)	Boolean	要求元素满足给定的最大长度规则
rangelength(range)	Boolean	要求元素满足给定的长度范围规则
min(value)	Boolean	要求元素满足给定的最小值规则
max(value)	Boolean	要求元素满足给定的最大值规则
range(range)	Boolean	要求元素满足给定值的范围规则
email( )	Boolean	要求元素满足 E-mail 格式规则
url( )	Boolean	要求元素满足 url 格式规则
date( )	Boolean	要求元素满足 date 格式规则
dateISO( )	Boolean	要求元素满足 ISO date 格式规则
dateDE( )	Boolean	要求元素满足 german date 格式规则
number( )	Boolean	要求元素满足带小数点的数字格式规则
numberDE( )	Boolean	要求元素满足 german format 并带小数点的数字格式规则
digits( )	Boolean	要求元素满足整型格式规则
creditcard( )	Boolean	要求元素满足信用卡号码格式规则
accept(extension)	Boolean	要求元素满足特定的文件格式
equalTo(other)	Boolean	要求元素等于另外一个元素
phoneUS( )	Boolean	要求元素满足美国电话号码的格式规则

■ Validator

Validation 验证会返回一个 Validator 对象，validator 对象可以帮助用户触发 validation 程序或者改变 form 的内容。validator 对象更多的方法如表 F-3 所示。

表 F-3　　validator 对象的方法

名　称	返回类型	说　明
form( )	Boolean	验证 form，如果验证合法则返回 true，否则返回 false
element(element)	Boolean	验证一个元素，如果验证合法则返回 true，否则返回 false
resetForm( )	undefined	复位被验证的 form
showErrors(errors)	undefined	显示指定的提示信息
numberOfInvalids()	Integer	返回无效字段的个数

validator 对象中的静态方法如表 F-4 所示。

表 F-4 validator 对象中的静态方法

名　称	返回类型	说　明
setDefaults(defaults)	undefined	修改 validation 初始的设置
addMethod(name, method, message)	undefined	增加一个新的 validation 方法。该方法必须由 name（必须是一个合法的 JavaScript 标识符）、一个基于函数的 JavaScript 和一个默认的字符串提示信息组成
addClassRules(name, rules)	undefined	增加一个验证规则，它代替了“rules”中的验证
addClassRules(rules)	undefined	增加多个验证规则

■ 实用项

表 F-5 实用项

名　称	返回类型	说　明
jQuery.format(template, argument, argumentN...)	String	使用参数来替换{n}占位符

■ 普通选择器

表 F-6 普通选择器

名　称	返回类型	说　明
:blank	Array<Element>	匹配值为空的元素
:filled	Array<Element>	匹配值不为空的元素
:unchecked	Array<Element>	匹配所有没被选择的元素

F.2 Form 插件 API

（1）Form 插件拥有很多方法，这些方法可以使用户很容易地管理表单数据和提交表单。

■ ajaxForm()

增加所需要的事件监听器，为 Ajax 提交表单做好准备。AjaxForm()方法并没有提交表单，而是在$(document).ready()方法中，使用 ajaxForm()方法来为 Ajax 提交表单做好准备。ajaxForm 方法可以接受 0 个或 1 个参数。单个的参数既可以是一个回调函数，也可以是一个 Options 对象。此方法可以进行链式操作。

例子：

```
$('#myFormId').ajaxForm();
```

■ ajaxSubmit()

立即通过 Ajax 方式提交表单。在大多数情况下，都是调用 ajaxSubmit()方法来响应用户的提交表单操作。AjaxSubmit()方法可以接受 0 个或 1 个参数。单个的参数既可以是一个回调函数，也可以是一个 Options 对象。此方法可以进行链式操作。

例子：

```
//绑定表单提交事件处理器
$('#myFormId').submit(function() {
    //提交表单。
    $(this).ajaxSubmit();
    //返回 false 以防止浏览器的提交
    return false;
});
```

■ formSerialize()

该方法将表单中所有的元素串行化（序列化）为一个字符串。formSerialize()方法会返回一个格式化好的字符串，格式如下：

```
name1=value1&name2=value2
```

因为返回的是字符串，而不是 jQuery 对象，所以该方法不能进行链式操作。

例子：

```
var queryString = $('#myFormId').formSerialize();

//然后可以使用$.get()、$.post()、$.ajax()等 Ajax 方法来提交数据
$.post('myscript.php', queryString , function(data){ });
```

■ fieldSerialize()

fieldSerialize()方法将表单的字段元素串行化（序列化）成一个字符串。当用户只需要串行化表单的一部分时就可以用到该方法了。fieldSerialize()方法会返回一个格式化后的字符串，格式如下：

```
name1=value1&name2=value2
```

因为返回的是字符串，所以该方法不可以进行链式操作。

例子：

```
var queryString = $('#myFormId .specialFields').fieldSerialize();
//将 id 为 myFormId 表单内 class 为 specialFields 的元素序列化
```

■ fieldValue()

fieldValue()方法把匹配元素的值插入到数组中，然后返回这个数组。从 0.91 版本起，该方法总是以数组的形式返回数据，如果元素值被判定无效，则数组为空，否则数组将包含一个或多个元素值。fieldValue()方法返回一个数组，因此不可以进行链式操作。

例子：

```
//取得密码框输入值
var value = $('#myFormId :password').fieldValue();
alert('第一个密码为: ' + value[0]);
```

■ resetForm()

该方法通过调用表单元素原有的 DOM 方法重置表单到初始状态。resetForm()方法可以进行链式操作。

例子：

```
$('#myFormId').resetForm();
```

■ clearForm()

clearForm()方法用来清空表单中的元素。该方法将所有的文本框（text）、密码框（password）和文本域（textarea）元素置空，清除下拉框（select）元素的选定以及将所有的单选按钮（radio）和多选按钮（checkbox）重置为非选定状态。clearForm()方法可以进行链式操作。

例子：

```
$('#myFormId').clearForm();
```

■ clearFields()

clearFields()方法用来清空字段元素。当用户需要清空一部分表单元素时就会用到该方法。clearFields()方法可以进行链式操作。

例子：

```
$('#myFormId .specialFields').clearFields();
//将 id 为 myFormId 表单内 class 为 specialFields 的元素清空
```

（2）Options 对象

ajaxForm()方法和 ajaxSubmit()方法支持许多选项，这些选项都可以通过 Options 对象来设置。Options 对象是一个简单的 JavaScript 对象，包含了如下属性与值的集合。

■ target

指明页面中根据服务器响应进行更新的元素。这个值可能是一个特殊的 jQuery 选择器字符串、一个 jQuery 对象或者一个 DOM 元素。

默认值：null。

■ url

将表单元素提交到指定的 url 中。

默认值：表单 action 属性的值。

■ type

指定提交表单数据的方法（method）：GET 或 POST。

默认值：表单 method 属性的值（如果没有找到，则为 GET）。

■ beforeSubmit

表单提交前的回调函数。beforeSubmit 回调函数被用来运行预提交逻辑或者校验表单数据。假如 beforeSubmit 回调函数返回 false，则表单将不会被提交。beforeSubmit 回调函数有 3 个参数：数组形式的表单数据、jQuery 表单对象和传递给 ajaxForm()方法或 ajaxSubmit()方法的 Options 对象。表单数据数组遵循以下数据格式（json 类型）。

```
[ { name: 'username', value: 'jresig' }, { name: 'password', value: 'secret' } ]
```

默认值：null。

■ success

表单成功提交后调用的回调函数。假如 success 回调函数被指定，将在服务器返回响应后被调用。success 函数可以传回 responseText 或者 responseXML 的值（决定值的数据类型是 dataType 选项）

默认值：null。

■ dataType

期望的服务器响应的数据类型，可以是 null、xml、script 或者 json。dataType 提供了指定的方法以便控制服务器的响应。这个指定的方法将被直接地反映到 jQuery.httpData()方法中。dataType 支持以下格式。

- xml。如果 dataType 被指定为 xml，服务器返回内容将被作为 XML 来对待。同时，如果“success”回调函数被指定，responseXML 的值将会传递给回调函数。
- json。如果 dataType 被指定为 json，服务器返回内容将被执行，如果“success”回调函数被指定，返回的内容将会传递给回调函数。
- script。如果 dataType 被指定为 script，服务器返回内容将被放在全局环境中执行。

默认值：null。

■ semantic

是否需要定义为严格的语义格式。注意，普通的表单序列化要遵循的语义不能包括 type 属性为 image 的 input 元素。假如服务器有严格的语义要求，而表单也至少包含一个 type="image"元素的时候，那么必须设置 semantic 选项为 true。

默认值：false。

■ resetForm

表单是否在提交成功后被重置。

默认值：null。

■ clearForm

表单是否在提交成功后被清空。

默认值：null。

■ iframe

表单是否总是将服务器响应指向到一个 iframe。iframe 在文件上传时会很有用。

默认值：false。

Form 插件实例：

```
//准备选项对象
var options = {
      target:   '#divToUpdate',
      url:      comment.php',
      success:  function() {
          alert('谢谢你的评论!');
      }
};

//在 ajaxForm()方法中使用 options 对象
$('#myForm').ajaxForm(options);
```

注意，利用此 Options 对象，可以将值传给 jQuery 的$.ajax()方法。假如用户熟悉$.ajax()方法提供的 options 对象，那么可以利用它们来将 Options 对象传递给 ajaxForm()方法和 ajaxSubmit()方法。

F.3 Livequery 插件 API

API 的官方网站地址为：http://docs.jquery.com/Plugins/livequery

表 F-7 **Livequery 插件的 API**

名　称	功能说明	返回类型	例　子
livequery(type,fn)	为所有匹配元素绑定给定类型的事件	jQuery	`$("p").livequery("click",function(){ alert($(this).text() ); });`
livequery(matchedFn)	为每个匹配元素触发一个函数	jQuery	`$("p").livequery(function(){ $(this).addClass('newClass'); });`
livequery(matchedFn, unmatchedFn)	为每个匹配元素触发第 1 个函数，如果元素不再匹配，则触发第 2 个函数	jQuery	`$("p").livequery(function(){ $(this).addClass('newClass'); }, function() { $(this).removeClass('newClass'); });`

续表

名　称	功能说明	返回类型	例　子
expire( )	它将停止/终止所有 Livequery 插件和选择器的关联	jQuery	`$("p").expire();`
expire(type)	它将停止/终止所有 Livequery 插件和事件类型、选择器的关联	jQuery	`$("p").expire('click');`

expire(type, fn)	它将停止/终止所有 Livequery 插件和事件类型、选择器、函数操作的关联	jQuery	`$("p").expire("click", function(){`
			`  alert($(this).text() );`
			`});`

expire(matchedFn)	它将停止/终止所有 Livequery 插件和选择器、匹配函数操作的关联	jQuery	`$("p").expire(function(){`
			`  $(this).addClass('newClass');`
			`});`

expire(matchedFn, unmatchedFn)	它将停止/终止所有 Livequery 插件和选择器、匹配函数、不匹配函数操作的关联	jQuery	`$("p").expire(function(){`
			`  $(this).addClass('newClass');`
			`}, function() {`
			`  $(this).removeClass('newClass');`
			`});`