jQuery.Form插件使用文档

目录

[jQuery.Form插件使用文档 1](#_Toc376591842)

[1 概述 2](#_Toc376591843)

[2 快速入门 2](#_Toc376591844)

[2.1 步骤一 2](#_Toc376591845)

[2.2 步骤二 2](#_Toc376591846)

[2.3 步骤三 3](#_Toc376591847)

[3 API说明 4](#_Toc376591848)

[3.1 属性 4](#_Toc376591849)

[3.1.1 可选参数对象 4](#_Toc376591850)

[3.1.2 表单数据对象 7](#_Toc376591851)

[3.2 事件 7](#_Toc376591852)

[3.2.1 success 7](#_Toc376591853)

[3.2.2 onSubmit 8](#_Toc376591854)

[3.2.3 onBeforeLoad 9](#_Toc376591855)

[3.2.4 onLoad 9](#_Toc376591856)

[3.2.5 onLoadError 10](#_Toc376591857)

[3.3 方法 10](#_Toc376591858)

[3.3.1 form 10](#_Toc376591859)

[3.3.2 submit 11](#_Toc376591860)

[3.3.3 load 12](#_Toc376591861)

[3.3.4 clear 13](#_Toc376591862)

[3.3.5 reset 14](#_Toc376591863)

[3.3.6 validate 14](#_Toc376591864)

[3.3.7 getFormData 15](#_Toc376591865)

[3.3.8 getOptions 15](#_Toc376591866)

# 概述

jQuery Form 插件能够让你简洁的将以HTML形式提交的表单升级成采用AJAX技术提交的表单。 插件里面主方法为form, 能够从form表单里采集信息确定如何处理表单的提交过程。该方法都支持众多的可选参数，能够让你对表单里数据的提交做到完全的控制。这让采用AJAX方式提交一个表单的过程简单的不能再简单了！

# 快速入门

## 步骤一

**在页面里写一个表单，比如一个登入功能的页面：**

<form id="loginForm" action="login.ashx" method="post">

昵称: <input type="text" name="name" />

密码: <input type="password" name=" password " />

<input type="submit" value="登入" />

</form>

## 步骤二

**引入jQuery和jQuery.form.js插件脚本，并在文档加载完成后初始化表单**

<head>

<title>登入</title>

<script src="Scripts/jquery.js"></script>

<script src="Scripts/jquery.form.js"></script>

<script>

$(function () {

//绑定loginForm表单并传入一个回调函数

$("#loginForm").form(function () {

alert("登入成功！");

});

});

</script>

</head>

这样就行了，当用户提交表单，如果服务器返回成功状态，那用户就可以看到”登入成功”的提示消息

## 步骤三

**更多操作**

* 通过可选参数绑定form元素

$("#loginForm").from({

url: "login.ashx",

dataType: "text", //提交成功后返回的数据类型

type: "POST", //请求方式

success: function (resultData) {

alert(resultData);//登入成功后显示返回的成功提示

}

});

* 方法的调用

本插件方法调用方式为：$("form").form("methodName",args)

* **例1**:

$("#loginForm").form("validate");//调用validate进行表单验证。

* **例2**：

$("#loginForm").form("submit", function (resultMsg) {

alert(resultMsg)

});

调用submit方法进行表单提交，成功后显示返回的成功提示

* 事件监听

本插件事件监听方式为：$("form").form({eventName:handleFunction})

* **例1**：

$("#loginForm").form({

onSubmit: function (requestFormData) {

//验证表单返回false阻止表单提交

return $("#loginForm").form("validate");

},

success: function (resultData) {

alert(resultData)

}

})

监听了onSubmit事件，该事件在表单提交前触发，返回false可以阻止表单提交。监听了success事件，当表单提交成功后触发该事件。

# API说明

## 属性

### 可选参数对象

form 和 submit 都支持大量的可选参数，它们通过可选参数项对象传入。可选参数项对象只是一个简单的 JavaScript对象，里边包含了一些属性和要监听的事件。它们的默认值定义在全局变量**$.fn.form.defaults**中，我们可以通过修改**$.fn.form.defaults**来改变默认值。*(注意：jQuery中 $.ajax方法使用到的可选参数同样适用于本可选参数对象)*可选参数项如下：

##### url

表单提交的地址。默认值为表单的action的值

##### dataType

预期服务器返回的数据类型，默认为”text“

##### type

表单提交的方式，'GET' 或 'POST'.

##### validateRules

一个验证规则对象集合，方法[validate](#_validate) 就是使用该对象来检验表单域的输入项是否正确。每个验证规则必须定义一个验证函数validator和验证无效时的提示函数message，例如，定义一个验证最小长度的规则：

$("form").form({

validateRules: {

minLength: {// //定义一个名称为minLength的验证规则

validator: function (value, params, $input) {

return value.length >= params[0];

},

message: function (value, params, $input) {

alert"请输入至少”+ params[0]+“个字符");

}

}

}

});

我们也可以通过修改**$.fn.form.defaults**来添加一个规则，效果是一样的：

$.fn.form.defaults.validateRules.minLength = {

validator: function (value, params, $input) {

return value.length >= params[0];

},

message: function (value, params, $input) {

alert("请输入至少"+ params[0]+"个字符");

}

};

validator和message有相同的三个参数value, params, $input，value表示表单域的值，params附加参数，$input是当前验证的表单域jQuery对象。如minLength[5]的params附加参数就是数组[5]，你也可以在这个数组中添加更多的参数，如minLength[5,”#output”].

现在你可以使用minlength来定义一个至少要输入5个字符的表单域：

<input type="text" name="nickname" validtype="minLength[5]" />

如果一个表单域要应用多个规则，那么多个规则之间用”|“分割：

<input type="text" name="fieldName"

validtype="minLength[10]|required|email" />

##### mappingRules

一个javascript对象，描述哪些表单域应用了哪些验证规则。

该对象的结构如下：

{

fieldName1:"validateRuleName1",

fieldName2:"validateRuleName2",

fieldName3:"validateRuleName3"

...

}

我们可以把验证规则直接写在html代码里：

<form action="/" method="post">

<input type="text" name="nickname"

validtype="minLength[20]" />

<input type="text" name="age" validtype="number" />

<input type="text" name="email" validtype="email" />

</form>

这里假设我们已经定义好了名称为minLength，number，email的验证规则。

更好的方式，我们还可以通过代码的方式来应用验证规则：

<form action="/" method="post">

<input type="text" name="nickname"/>

<input type="text" name="age"/>

<input type="text" name="email"/>

</form>

然后添加代码：

$("form").form({

mappingRules:{

nickname:"minLength[20]",

age: "number",

email:"email"

},

validateRules: {

minLength: {

validator: function (value, params, $input) {

return value.length >= params[0];

},

message: function (value, params, $input) {

alert("请输入至少"+ params[0]+"个字符");

}

},

number: {

validator: function (value, params, $input) {

return /^\d+$/.test(value);

},

message: function (value, params, $input) {

alert("请输入一个整数");

}

},

email:{

validator: function (value, params, $input) {

var r = /^([a-z0-9\_\.-]+)@([\da-z\.-]+)\.([a-z\.]{2,6})$/;

return r.test(value);

},

message: function (value, params, $input) {

alert("请输入一个正确的email地址");

}

}

},

onSubmit: function () {

return $("form").form("validate");

},

});

##### 一些事件

onSubmit

success

onBeforeLoad

onLoad

onLoadError

以上事件字段请参见[事件](#_事件)

### 表单数据对象

本文档所说的表单数据或表单数据对象或formData说的就是键名为表单域的名称对应的值为表单域的值 所组成的javascript的对象，它的结构如下：

{

fieldName1:val1,

fieldName2:val2,

fieldName3:val3,

...

}

例如下面的表单：

<form >

<input type="text" name="nickname" value="peter"/>

<input type="text" name="age" value="20"/>

<input type="text" name="email" value="peter@xxx.com"/>

</form>

它的表单数据对象就为：

var formData = {

nickname: "peter",

age: 20,

email: [peter@xxx.com](mailto:peter@xxx.com)

}

## 事件

### success

* 描述：

{

success: function (resultData) {

// resultData 为表单提交成功时服务端的返回值

}

}

当表单提交后触发该事件，如果'success' 回调函数被指定，当表单提交成功后，该回调行数就会被执行，同时服务端的返回值就会传递给回调函数(可以通过dataType可选参数指定返回值类型)

* .示例

$("#loginForm").from({

url: "login.ashx",

dataType: "json", //提交成功后返回的数据类型

type: "POST", //请求方式

success: function (data) {//假设json格式为：{status:1,msg: "登入成"}

if (data.status == 1) {

alert(data.msg);

}

}

});

### onSubmit

* 描述：

{

onSubmit: function (formData) {

// formData 为表单数据对象

}

}

表单将要提交前触发该事件，这个可以用在表单提交前的预处理，或表单校验，如果'onSubmit'指定的函数返回false，则表单不会被提交。该函数有个参数，它的值为将要提交的表单数据对象，格式为：{fieldName:fieldValue,...}

* .示例

$("#loginForm").form({

onSubmit: function (formData) {

//表单提交前添加一个dataTime参数

formData.dataTime = "2013-12-12 12:12";

for (var name in formData) {//弹出表单的所有值

alert(name + " : " + formData[name]);

}

return $("#loginForm").form("validate");

},

success: function (msg) {

alert(msg);

}

})

### onBeforeLoad

* 描述：

{

onBeforeLoad: function (param) {

// param 请求远程数据的附加参数

}

}

在发送载入远程数据请求之前触发，这个可以用在载入表单数据前预设一些附加的请求参数，如果' onBeforeLoad '指定的函数返回false，则表单不会载入数据。

*注：只有调用了"load"方法才会触发onBeforeLoad，onLoad，onLoadError这些事件。*

* 示例

$("#loginForm").form({

onBeforeLoad: function (param) {

param.name = "peter";

}

}).form("load", "/userInfo.ashx");

将用户名称为peter的用户信息载入表单。

### onLoad

* 描述：

{

 onLoad: function (formData) {

// formData 将要填充的表单的表单数据。

}

}

在远程数据载入但没填充表单之前，此时对formData数据做一些修改等操作。

* 示例

$("#loginForm").form({

onLoad: function (formData) {

formData.price = "￥"+formData.price;

}

}).form("load", "/books.ashx");

在书本信息数据填充表单之前，改变价格的显示的格式。

### onLoadError

* 描述：

{

onLoadError: function (errorMsg) {

// errorMsg 错误信息

}

}

在载入远程表单数据发生错误时触发

* 示例

$("#loginForm").form({

onLoadError: function (errorMessage) {

alert(errorMessage);

}

}).form("load", "/books.ashx");

如果在载入远程表单数据发生错误，则弹出错误提示对话框

## 方法

### form

* 说明

$("form").form([methodName | options | callback], [args…]);

本插件中form方法是唯一绑定在jQuery对象上的方法，我们称form方法为主方法。其他方法的执行都用通过主方法form来实现。

Form方法的第一个参数可以是一个方法名称methodName或一个可选参数对象options或一个回调函数callback。当第一个参数为方法名称时，则后面的所有参数args就为传给该方法的参数。当第一个参数不是方法名称时，后面的所以参数agrs将无效。

* 参数
* methodName

要执行的方法名称

* options

[可选参数对象](#_可选参数对象)

* callback

表单提交成功时的回调函数

* 示例
* 例1

$("#userInfoForm").form("validate");//验证用户信息

$("#userInfoForm").form("load", "/users.ashx", "post");//加载用户信息

主方法的参数为方法名称。

* 例2

$("form").form({

url: "Handler.ashx",

dataType: "text",

type: "POST",

onSubmit: function (formData) {//提交前验证

return $form.form("validate");

},

success: function (msg) {//提交成功后显示成功提示语。

alert(msg)

}

});

主方法的参数为[可选参数对象](#_可选参数对象)。

* 例3

$("#loginForm").form(function (msg) {//提交成功后显示成功提示语。

alert(msg)

});

主方法的参数为回调函数

### submit

* 说明

$("form").form("submit", [options | callback]);

通过AJAX方式提交表单. submit方法需要零个或一个参数，唯一的一个参数可以是一个回调函数或者是一个可选参数对象。

* 参数
* options

可选参数对象

* callback

提交成功后的回调函数。

* 返回值

返回匹配的jQuery对象集合

* 示例
* 例1：

$("#loginForm").form("submit", function (msg) {

alert(msg)

});

立即提交表单，如果成功则显示成功信息。本例参数为回调函数。

* 例2

$("#loginForm").form("submit",{

onSubmit: function (formData) {

return $("#loginForm").form("validate");

},

success: function (msg) {

alert(msg);

}

})

立即提交表单，如果验证成功顺利提交表单则显示成功信息。本例参数为可选参数对象。

### load

* 说明

$("form").form("load", [formData | uri ],[type]);

载入数据填充表单，load方法可以传入一个或两个参数，第一个参数可以是一个uri字符串或者是一个表单数据对象类型。第二个参数只有第一个参数为uri时才有效，它是请求远程数据时的请求类型,有get和post可选，默认为post。

在这些load函数时会触发onBeforeLoad，onLoad，onLoadError事件，您可以监听这些事件来做一些相关的处理。

* 参数
* formData

表单数据对象

* uri

请求表单数据对象的远程地址

* type

请求类型，有get和post可选，默认为post

* 返回值

返回匹配的jQuery对象集合

* 示例
* 例1

$("#userInfoForm").form({

onBeforeLoad: function (params) {

params.id = 110;//添加请求参数

}

});

$("#userInfoForm").form("load", "/users.ashx", "post");

加载用户id为110的用户信息，并在表单中显示。也可以这样实现：

$("#userInfoForm").form("load", "/users.ashx?id=110", "post");

* 例2

var userInfo = {

username:"peter",

age:20,

address:"guanghangong"

}

$("#userInfoForm").form("load", userInfo);

直接用表单数据对象填充表单。

### clear

* 说明

$("form").form("clear", [defaultValues]);

清空表单所有元素的值，clear方法可以有0个或一个参数，唯一的一个参数为一个默认值对象，它的作用是等表单清空后使用它所指的值来设置表单中各个表单域的值。

* 参数
* defaultValues

表单域的默认值集合，其格式为：

{

fieldName1:val1,

fieldName2:val2,

fieldName3:val3,

...

}

* 返回值

返回匹配的jQuery对象集合

* 示例
* 例1

$(" #userInfoForm ").form("clear");

将会清空所有的文本框，密码框，文本域里的值，去掉下拉列表所有被选中的项，让所有复选框和单选框里被选中的项不再选中。

* 例2

$("#userInfoForm").form("clear", {

username:"peter",

age:20

});

清空用户表单所有元素的值，并设置nane为age的input表单域为20，设置nane为username的input表单域为peter

### reset

* 说明

$("form").form("reset");

通过调用表单元素的内在的DOM 上的方法把表单重置成最初的状态，reset方法无参数

* 参数

无

* 返回值

返回匹配的jQuery对象集合

* 示例
* 例1

$(" #userInfoForm ").form("reset");

重置用户信息。

### validate

* 说明

$("form").form("validate", [showAllError]);

执行表单域验证，如果全部表单域有效返回true，否则返回false。validate方法可以有0个或一个参数，唯一的一个参数为bool类型表示如果表单域验证返回false那么是否要显示所有表单域的错误。

validate方法使用的验证规则定义在$.fn.form.defaults.validateRules

* 参数
* showAllError

bool类型，表示是否要显示所有表单域的错误。如果showAllError

为false则只显示第一个出现的错误

* 示例
* 例1

$form.form({

onSubmit: function (requestFormData) {

return $form.form("validate");

},

success: function (msg) {

alert(msg)

},

validateRules: {

//定义一个minLength规则，表示被绑定的字段的字符串长度要不小于6个字符

minLength: {

validator: function (value, params, $input) {

return value.length >= params[0];

},

message: function (value, params, $input) {

alert("请输入 " + params[0] + " 个字符");

}

}

},

mappingRules: {

//将规则应用在名称为nickName的表单域

nickName: "minLength[6]"上

}

});

当表单提交时，会验证输入的nickName的长度是否会不小于6个字符，如果验证通过则提交表单，否则显示错误。

### getFormData

* 说明

$("form").form("getFormData");

获取表单数据。getFormData无参数。

* 参数

无

* 示例

无

### getOptions

* 说明

$("form").form("getOptions ");

获取绑定在表单上的可选参数对象。getOptions无参数。

* 参数

无

* 示例

无