**Jquery中的事件处理部分**

## **ready(fn)**

### 概述

当DOM载入就绪可以查询及操纵时绑定一个要执行的函数。

这是事件模块中最重要的一个函数，因为它可以极大地提高web应用程序的响应速度。

简单地说，这个方法纯粹是对向window.load事件注册事件的替代方法。通过使用这个方法，可以在DOM载入就绪能够读取并操纵时立即调用你所绑定的函数，而99.99%的JavaScript函数都需要在那一刻执行。

有一个参数－－对jQuery函数的引用－－会传递到这个ready事件处理函数中。可以给这个参数任意起一个名字，并因此可以不再担心命名冲突而放心地使用$别名。

请确保在 <body> 元素的onload事件中没有注册函数，否则不会触发+$(document).ready()事件。

可以在同一个页面中无限次地使用$(document).ready()事件。其中注册的函数会按照（代码中的）先后顺序依次执行。

### 参数

#### ****fn****FunctionV1.0

要在DOM就绪时执行的函数

### 示例

#### 描述:

在DOM加载完成时运行的代码，可以这样写：

##### jQuery 代码:

$(document).ready(function(){

// 在这里写你的代码...

});

#### 描述:

使用 $(document).ready() 的简写，同时内部的 jQuery 代码依然使用 $ 作为别名，而不管全局的 $ 为何。

##### jQuery 代码:

$(function($) {

// 你可以在这里继续使用$作为别名...

});

## **on(events,[selector],[data],fn)**

### 概述

在选择元素上绑定一个或多个事件的事件处理函数。

on()方法绑定事件处理程序到当前选定的jQuery对象中的元素。在jQuery 1.7中，.on()方法 提供绑定事件处理程序所需的所有功能。帮助从旧的jQuery事件方法转换，see [.bind()](http://api.jquery.com/bind/), [.delegate()](http://api.jquery.com/delegate/), 和 [.live()](http://api.jquery.com/live/). 要删除的.on()绑定的事件，请参阅[.off()](http://api.jquery.com/off/)。要附加一个事件，只运行一次，然后删除自己， 请参阅[.one()](http://api.jquery.com/one/)

### 参数

#### ****events,[selector],[data],fn****V1.7

****events****:一个或多个用空格分隔的事件类型和可选的命名空间，如"click"或"keydown.myPlugin" 。

****selector****:一个选择器字符串用于过滤器的触发事件的选择器元素的后代。如果选择的< null或省略，当它到达选定的元素，事件总是触发。

****data****:当一个事件被触发时要传递event.data给事件处理函数。

****fn****:该事件被触发时执行的函数。 false 值也可以做一个函数的简写，返回false。

#### ****events-map,[selector],[data]****V1.7

****events-map****:个用字符串表示的，一个或多个空格分隔的事件类型和可选的命名空间，值表示事件绑定的处理函数。

****selector****:一个选择器字符串过滤选定的元素，该选择器的后裔元素将调用处理程序。如果选择是空或被忽略，当它到达选定的元素，事件总是触发。

****data****:当一个事件被触发时要传递event.data给事件处理函数。

### 示例

#### 描述:

Display a paragraph's text in an alert when it is clicked:

$("p").on("click", function(){

alert( $(this).text() );

});

Pass data to the event handler, which is specified here by name:

function myHandler(event) {

alert(event.data.foo);

}

$("p").on("click", {foo: "bar"}, myHandler)

Cancel a form submit action and prevent the event from bubbling up by returning false:

$("form").on("submit", false)

Cancel only the default action by using .preventDefault().

$("form").on("submit", function(event) {

event.preventDefault();

});

Stop submit events from bubbling without preventing form submit, using .stopPropagation().

$("form").on("submit", function(event) {

event.stopPropagation();

});

## **off(events,[selector],[fn])**

### 概述

在选择元素上移除一个或多个事件的事件处理函数。

off() 方法移除用[.on()](http://api.jquery.com/on)绑定的事件处理程序。有关详细信息，请参阅该网页上delegated和directly绑定事件。特定的事件处理程序可以被移除元素上提供事件的名称，命名空间，选择器，或处理函数名称的组合。****当有多个过滤参数，所提供的参数都必须匹配的事件处理程序被删除。****

如果一个简单的事件名称，比如提供"click"，所有 这种类型的事件（包括直接和委派）从jQuery设置的元素上删除。当编写代码，将作为一个插件使用，或者干脆当一个大的代码基础工作，最好的做法是安装和取下使用命名空间的事件，从而使代码不会无意中删除其他代码附加事件处理程序。在一个特定的命名空间中的所有类型的所有事件，可以从一个元素中删除，只是提供了一个命名空间，比如 ".myPlugin"。至少，无论是命名空间或事件名称必须提供。

要删除特定的委派事件处理程序，提供一个selector 的参数。选择器字符串必须是完全匹配递到.on()事件处理程序附加的选择器。要删除非委托元素上的所有事件，使用特殊值 "\*\*" 。

处理程序也可以删除handler参数指定名称的函数。当jQuery的绑定一个事件处理程序，它分配一个唯一的ID给处理函数。函数用[jQuery.proxy()](http://api.jquery.com/jQuery.proxy)代理或类似有相同的唯一ID机制（代理函数），因此，通过代理处理程序.off 可能会删除比预期更多的处理程序。在这些情况下，最好是附加和移除事件处理程序，使用命名空间。

和.on()一样，你可以传递一个 events-map>参数明确的指定而不是用events 和 handler作为单独参数。键事件和/或命名空间;值是处理函数或为false的特殊价值。

### 参数

#### ****events,[selector],[fn]****V1.7

****events****:一个或多个空格分隔的事件类型和可选的命名空间，或仅仅是命名空间，比如"click", "keydown.myPlugin", 或者 ".myPlugin".

****selector****:一个最初传递到.on()事件处理程序附加的选择器。

****fn****:事件处理程序函数以前附加事件上，或特殊值false.

#### ****events-map,[selector]****V1.7

****events-map****:一个用字符串表示的，一个或多个空格分隔的事件类型和可选的命名空间，值表示先前事件绑定的处理函数。

****selector****:一个最初传递到.on()事件处理程序附加的选择器。

### 示例

#### 描述:

Remove all event handlers from all paragraphs:

$("p").off()

Remove all delegated click handlers from all paragraphs:

$("p").off( "click", "\*\*" )

Remove just one previously bound handler by passing it as the third argument:

var foo = function () {

// code to handle some kind of event

};

// ... now foo will be called when paragraphs are clicked ...

$("body").on("click", "p", foo);

// ... foo will no longer be called.

$("body").off("click", "p", foo);

Unbind all delegated event handlers by their namespace:

var validate = function () {

// code to validate form entries

};

// delegate events under the ".validator" namespace

$("form").on("click.validator", "button", validate);

$("form").on("keypress.validator", "input[type='text']", validate);

// remove event handlers in the ".validator" namespace

$("form").off(".validator");

## **bind(type,[data],fn)**

### 概述

为每个匹配元素的特定事件绑定事件处理函数。

### 参数

#### ****type,[data],function(eventObject)****String,Object,FunctionV1.0

****type:**** 含有一个或多个事件类型的字符串，由空格分隔多个事件。比如"click"或"submit"，还可以是自定义事件名。

****data:****作为event.data属性值传递给事件对象的额外数据对象

****fn:****绑定到每个匹配元素的事件上面的处理函数

#### ****type,[data],false****String,Object,boolV1.4.3

****type:****含有一个或多个事件类型的字符串，由空格分隔多个事件。比如"click"或"submit"，还可以是自定义事件名。

****data:****作为event.data属性值传递给事件对象的额外数据对象

****false:**** 将第三个参数设置为false会使默认的动作失效。

#### ****events****StringV1.4

一个或多个事件类型的字符串和函数的数据映射来执行他们。

### 示例

#### 描述:

当每个段落被点击的时候，弹出其文本。

##### jQuery 代码:

$("p").bind("click", function(){

alert( $(this).text() );

});

#### 描述:

同时绑定多个事件类型

##### jQuery 代码:

$('#foo').bind('mouseenter mouseleave', function() {

$(this).toggleClass('entered');

});

#### 描述:

同时绑定多个事件类型/处理程序

##### jQuery 代码:

$("button").bind({

click:function(){$("p").slideToggle();},

mouseover:function(){$("body").css("background-color","red");},

mouseout:function(){$("body").css("background-color","#FFFFFF");}

});

#### 描述:

你可以在事件处理之前传递一些附加的数据。

##### jQuery 代码:

function handler(event) {

alert(event.data.foo);

}

$("p").bind("click", {foo: "bar"}, handler)

#### 描述:

通过返回false来取消默认的行为并阻止事件起泡。

##### jQuery 代码:

$("form").bind("submit", function() { return false; })

#### 描述:

通过使用 preventDefault() 方法只取消默认的行为。

##### jQuery 代码:

$("form").bind("submit", function(event){

event.preventDefault();

});

#### 描述:

通过使用 stopPropagation() 方法只阻止一个事件起泡。

##### jQuery 代码:

$("form").bind("submit", function(event){

event.stopPropagation();

});

## **one(type,[data],fn)**

### 概述

为每一个匹配元素的特定事件（像click）绑定一个一次性的事件处理函数。

在每个对象上，这个事件处理函数只会被执行一次。其他规则与bind()函数相同。这个事件处理函数会接收到一个事件对象，可以通过它来阻止（浏览器）默认的行为。如果既想取消默认的行为，又想阻止事件起泡，这个事件处理函数必须返回false。

多数情况下，可以把事件处理函数定义为匿名函数（见示例一）。在不可能定义匿名函数的情况下，可以传递一个可选的数据对象作为第二个参数（而事件处理函数则作为第三个参数），见示例二。

### 参数

#### ****type,[data],fn****String,Object,FunctionV1.1

****type:****添加到元素的一个或多个事件。由空格分隔多个事件。必须是有效的事件。

****data:****将要传递给事件处理函数的数据映射

****fn:****每当事件触发时执行的函数。

### 示例

#### 描述:

当所有段落被第一次点击的时候，显示所有其文本。

##### jQuery 代码:

$("p").one("click", function(){

alert( $(this).text() );

});

## **trigger(type,[data])**

### 概述

在每一个匹配的元素上触发某类事件。

这个函数也会导致浏览器同名的默认行为的执行。比如，如果用trigger()触发一个'submit'，则同样会导致浏览器提交表单。如果要阻止这种默认行为，应返回false。

你也可以触发由bind()注册的自定义事件而不限于浏览器默认事件。

事件处理函数会收到一个修复的(规范化的)事件对象，但这个对象没有特定浏览器才有的属性，比如keyCode。

jQuery也支持 <a href="http://docs.jquery.com/Namespaced\_Events">命名空间事件</a>。这允许你触发或者解除绑定一组特定的事件处理函数，而无需一一个指定。你可以在事件类型后面加上感叹号 ! 来只触发那些没有命名空间的事件处理函数。

****jQuery 1.3中新增：****

所有触发的事件现在会冒泡到DOM树上了。举例来说，如果你在一个段落p上触发一个事件，他首先会在这个元素上触发，其次到父元素，在到父元素的父元素，直到触发到document对象。这个事件对象有一个 .target 属性指向最开始触发这个事件的元素。你可以用 stopPropagation() 来阻止事件冒泡，或者在事件处理函数中返回false即可。

事件对象构造器现在已经公开，并且你可以自行创建一个事件对象。这个事件对象可以直接传递给trigger所触发的事件处理函数。事件对象的完整属性列表可以在 <a href="http://docs.jquery.com/Events/jQuery.Event">jQuery.Event</a> 的文档里找到。

你可以有三种方式指定事件类型：

\* 你可以传递字符串型的事件名称(type参数)。

\* 你可以使用jQuery.Event对象。可以将数据放进这个对象，并且这个对象可以被触发的事件处理函数获取到。

\* 最后，你可以传递一个带有数据的字面量对象。他将被复制到真正的jQuery.Event对象上去。 注意在这种情况下你'''必须'''指定一个 <em>type</em> 属性。

### 参数

#### ****type,[data]****String|Event,ArrayV1.0

****type:****一个事件对象或者要触发的事件类型

****data:****传递给事件处理函数的附加参数

#### ****event****ObjectV1.3

事件发生时运行的函数

### 示例

#### 描述:

提交第一个表单，但不用submit()

##### jQuery 代码:

$("form:first").trigger("submit")

#### 描述:

给一个事件传递参数

##### jQuery 代码:

$("p").click( function (event, a, b) {

// 一个普通的点击事件时，a和b是undefined类型

// 如果用下面的语句触发，那么a指向"foo",而b指向"bar"

} ).trigger("click", ["foo", "bar"]);

#### 描述:

下面的代码可以显示一个"Hello World"

##### jQuery 代码:

$("p").bind("myEvent", function (event, message1, message2) {

alert(message1 + ' ' + message2);

});

$("p").trigger("myEvent", ["Hello","World!"]);

## **unbind(type,[data|fn]])**

### 概述

bind()的反向操作，从每一个匹配的元素中删除绑定的事件。

### 参数

#### ****type,[fn]****String,FunctionV1.0

****type:****删除元素的一个或多个事件,由空格分隔多个事件值。

****fn:****要从每个匹配元素的事件中反绑定的事件处理函数

#### ****type,false****String,boolV1.4.3

****type:****删除元素的一个或多个事件,由空格分隔多个事件值

****false:****设置为false会使默认的动作失效。

#### ****eventObj****StringV1.0

事件对象。这个 eventObj 参数来自事件绑定函数

### 示例

#### 描述:

把所有段落的所有事件取消绑定

##### jQuery 代码:

$("p").unbind()

#### 描述:

将段落的click事件取消绑定

##### jQuery 代码:

$("p").unbind( "click" )

#### 描述:

删除特定函数的绑定，将函数作为第二个参数传入

##### jQuery 代码:

var foo = function () {

// 处理某个事件的代码

};

$("p").bind("click", foo); // ... 当点击段落的时候会触发 foo

$("p").unbind("click", foo); // ... 再也不会被触发 foo

## **live(type, [data], fn)**

### 概述

jQuery 给所有匹配的元素附加一个事件处理函数，即使这个元素是以后再添加进来的也有效。

这个方法是基本是的 .bind() 方法的一个变体。使用 .bind() 时，选择器匹配的元素会附加一个事件处理函数，而以后再添加的元素则不会有。为此需要再使用一次 .bind() 才行。比如说

<body>

<div class="clickme">Click here</div>

</body>

可以给这个元素绑定一个简单的click事件：

$('.clickme').bind('click', function() {

alert("Bound handler called.");

});

当点击了元素，就会弹出一个警告框。然后，想象一下这之后有另一个元素添加进来了。

$('body').append('<div class="clickme">Another target</div>');

尽管这个新的元素也能够匹配选择器 ".clickme" ，但是由于这个元素是在调用 .bind() 之后添加的，所以点击这个元素不会有任何效果。

.live() 就提供了对应这种情况的方法。如果我们是这样绑定click事件的：

$('.clickme').live('click', function() {

alert("Live handler called.");

});

然后再添加一个新元素：

$('body').append('<div class="clickme">Another target</div>');

然后再点击新增的元素，他依然能够触发事件处理函数。

****事件委托****

.live() 方法能对一个还没有添加进DOM的元素有效，是由于使用了事件委托：绑定在祖先元素上的事件处理函数可以对在后代上触发的事件作出回应。传递给 .live() 的事件处理函数不会绑定在元素上，而是把他作为一个特殊的事件处理函数，绑定在 DOM 树的根节点上。在我们的例子中，当点击新的元素后，会依次发生下列步骤：

1. 生成一个click事件传递给

来处理

1. 由于没有事件处理函数直接绑定在 <divgt; 上，所以事件冒泡到DOM树上
2. 事件不断冒泡一直到DOM树的根节点，默认情况下上面绑定了这个特殊的事件处理函数。
3. 执行由 .live() 绑定的特殊的 click 事件处理函数。
4. 这个事件处理函数首先检测事件对象的 target 来确定是不是需要继续。这个测试是通过检测 $(event.target).closest('.clickme') 能否找到匹配的元素来实现的。
5. 如果找到了匹配的元素，那么调用原始的事件处理函数。

由于只有在事件发生时才会在上面的第五步里做测试，因此在任何时候添加的元素都能够响应这个事件。

****附加说明****

.live() 虽然很有用，但由于其特殊的实现方式，所以不能简单的在任何情况下替换 .bind()。主要的不同有：

* 在jQuery 1.4中，.live()方法支持自定义事件，也支持所有的 JavaScript 事件。在jQuery 1.4.1中，甚至也支持 focus 和 blue 事件了（映射到更合适，并且可以冒泡的focusin和focusout上）。另外，在jQuery 1.4.1中，也能支持hover（映射到"mouseenter mouseleave"）。然而在jQuery 1.3.x中，只支持支持的JavaScript事件和自定义事件：click, dblclick, keydown, keypress, keyup, mousedown, mousemove, mouseout, mouseover, 和 mouseup.
* .live() 并不完全支持通过DOM遍历的方法找到的元素。取而代之的是，应当总是在一个选择器后面直接使用 .live() 方法，正如前面例子里提到的。
* 当一个事件处理函数用 .live() 绑定后，要停止执行其他的事件处理函数，那么这个函数必须返回 false。 仅仅调用 .stopPropagation() 无法实现这个目的。

参考 .bind() 方法可以获得更多关于事件绑定的信息。

在jQuery 1.4.1 中，.live() 能接受多个，用空间分隔事件，在提供类似.bind()的功能 。例如，我们可以“live ” 同时绑定mouseover和mouseout事件,像这样：

$('.hoverme').live('mouseover mouseout', function(event) {

if (event.type == 'mouseover') {

// do something on mouseover

} else {

// do something on mouseout

}

});

在jQuery 1.4.3中:你可以绑定一个或多个事件类型的字符串和函数的数据映射来执行他们

$("a").live({

click: function() {

// do something on click

},

mouseover: function() {

// do something on mouseover

}

});

在jQuery 1.4 中，data参数可以用于把附加信息传递给事件处理函数。一个很好的用处是应付由闭包导致的问题。可以参考 .bind() 的讨论来获得更多信息。

在jQuery 1.4 中， live事件可以绑定到“context”DOM元素，而不是默认的文档的根。要设置此背景下，我们通过在一个单一的DOM元素(而不是一个jQuery集合或选择器)使用jQuery() function's second argument。

$('div.clickme', $('#container')[0]).live('click', function() {

// Live handler called.

});

### 参数

#### ****type,[fn]****String,FunctionV1.3

****type:****一个或多个事件类型，由空格分隔多个事件。

****fn:****要从每个匹配元素的事件中反绑定的事件处理函数

#### ****type,[data],false****String,Array,boolV1.4

****type:****一个或多个事件类型，由空格分隔多个事件。

****data:****传递给事件处理函数的附加参数

****false:****设置为false会使默认的动作失效。

#### ****event****StringV1.4.3

一个或多个事件类型的字符串和函数的数据映射来执行他们

### 示例

#### 描述:

点击生成的p依然据有同样的功能。

##### HTML 代码:

<p>Click me!</p>

##### jQuery 代码:

$("p").live("click", function(){

$(this).after("<p>Another paragraph!</p>");

});

#### 描述:

阻止默认事件行为和事件冒泡，返回 false

##### jQuery 代码:

$("a").live("click", function() { return false; });

#### 描述:

仅仅阻止默认事件行为

##### jQuery 代码:

$("a").live("click", function(event){

event.preventDefault();

});

## **die(type, [fn])**

### 概述

从元素中删除先前用.live()绑定的所有事件.(此方法与live正好完全相反。)

如果不带参数，则所有绑定的live事件都会被移除。

你可以解除用live注册的自定义事件。

如果提供了type参数，那么会移除对应的live事件。

如果也指定了第二个参数function,则只移出指定的事件处理函数。

### 参数

#### ****type[,fn]****String,FunctionV1.3

****type:****要移除的一个或多个事件处理程序。由空格分隔多个事件值。必须是有效的事件。

****fn:****要移除的函数。。

#### ****type****StringV1.4.3

要移除的一个或多个事件处理程序。 由空格分隔多个事件值。必须是有效的事件。

### 示例

#### 描述:

给按钮解除click事件

##### jQuery 代码:

function aClick() {

$("div").show().fadeOut("slow");

}

$("#unbind").click(function () {

$("#theone").die("click", aClick)

});

## **delegate(selector,[type],[data],fn)**

### 概述

指定的元素（属于被选元素的子元素）添加一个或多个事件处理程序，并规定当这些事件发生时运行的函数。

### 参数

#### ****selector,[type],fn****String,String,FunctionV1.4.2

****selector:****选择器字符串，用于过滤器触发事件的元素。

****type:****附加到元素的一个或多个事件。 由空格分隔多个事件值。必须是有效的事件。

****fn:****当事件发生时运行的函数

#### ****selector,[type],[data],fn****String,String,Object,FunctionV1.4.2

****selector:****选择器字符串，用于过滤器触发事件的元素。

****type:****附加到元素的一个或多个事件。由空格分隔多个事件值。必须是有效的事件。

****data:****传递到函数的额外数据

****fn:****当事件发生时运行的函数

#### ****selector,events****String,StringV1.4.3

****selector:****选择器字符串，用于过滤器触发事件的元素。

****events:****一个或多个事件类型的字符串和函数的数据映射来执行他们。

### 示例

#### 描述:

当点击鼠标时，隐藏或显示 p 元素：

##### HTML 代码:

<div style="background-color:red">

<p>这是一个段落。</p>

<button>请点击这里</button>

</div>

##### jQuery 代码:

$("div").delegate("button","click",function(){

$("p").slideToggle();

});

#### 描述: delegate这个方法可作为live()方法的替代，使得每次事件绑定到特定的DOM元素。

##### 以下两段代码是等同的:

$("table").delegate("td", "hover", function(){  
 $(this).toggleClass("hover");

});

$("table").each(function(){  
  $("td", this).live("hover", function(){  
   $(this).toggleClass("hover");  
  });

});

## **undelegate([sel,[type],fn])**

### 概述

删除由 delegate() 方法添加的一个或多个事件处理程序。

### 参数

#### ****selector,[type]****String,String1.4.2

****selector:****需要删除事件处理程序的选择器。

****type:****需要删除处理函数的一个或多个事件类型。 由空格分隔多个事件值。必须是有效的事件。

#### ****selector,[type],fn****String,String,FunctionV1.4.2

****selector:****需要删除事件处理程序的选择器。

****type:****需要删除处理函数的一个或多个事件类型。 由空格分隔多个事件值。必须是有效的事件。

****fn:****要删除的具体事件处理函数。

#### ****selector,events****String,StringV1.4.3

****selector:****需要删除事件处理程序的选择器。

****events:****一个或多个事件类型的字符串和函数的数据映射来执行他们。

#### ****namespace****String1.6

****namespace:****一个字符串，其中包含一个命名空间取消绑定的所有事件。

### 示例

#### 描述:

从p元素删除由 delegate() 方法添加的所有事件处理器：

##### jQuery 代码:

$("p").undelegate();

从p元素删除由 delegate() 方法添加的所有click事件处理器：

##### jQuery 代码:

$("p").undelegate( "click" )

从form元素删除由 delegate() 方法添加的".whatever"命名空间：

##### jQuery 代码:

var foo = function () {

//.....

};

//用delegate() 方法给click事件增加".whatever"命名空间

$("form").delegate(":button", "click.whatever", foo);

$("form").delegate("input[type='text']", "keypress.whatever", foo);

// 用undelegate()方法删除delegate()方法增加的".whatever"命名空间

$("form").undelegate(".whatever");

## **hover([over,]out)**

### 概述

一个模仿悬停事件（鼠标移动到一个对象上面及移出这个对象）的方法。这是一个自定义的方法，它为频繁使用的任务提供了一种“保持在其中”的状态。

当鼠标移动到一个匹配的元素上面时，会触发指定的第一个函数。当鼠标移出这个元素时，会触发指定的第二个函数。而且，会伴随着对鼠标是否仍然处在特定元素中的检测（例如，处在div中的图像），如果是，则会继续保持“悬停”状态，而不触发移出事件（修正了使用mouseout事件的一个常见错误）。

### 参数

#### ****over,out****Function,FunctionV1.0

over:鼠标移到元素上要触发的函数

out:鼠标移出元素要触发的函数

#### ****out****ObjectV1.4

当鼠标移到元素上或移出元素时触发执行的事件函数

### 示例

#### over,out 描述:

鼠标悬停的表格加上特定的类

##### jQuery 代码:

$("td").hover(

function () {

$(this).addClass("hover");

},

function () {

$(this).removeClass("hover");

}

);

#### out 描述:

hovre()方法通过绑定变量"handlerInOut"来切换方法。

##### jQuery 代码:

$("td").bind("mouseenter mouseleave",handlerInOut);

$("td").hover(handlerInOut);

## **toggle([speed],[easing],[fn])**

### 概述

用于绑定两个或多个事件处理器函数，以响应被选元素的轮流的 click 事件。

如果元素是可见的，切换为隐藏的；如果元素是隐藏的，切换为可见的。

### 参数

#### ****fn,fn2,[fn3,fn4,...]****Function,....V1.0

fn:第一数次点击时要执行的函数。

fn2:第二数次点击时要执行的函数。

fn3,fn4,...:更多次点击时要执行的函数。

#### ****[speed] [,fn]****String,FunctionV1.0

speed: 隐藏/显示 效果的速度。默认是 "0"毫秒。可能的值：slow，normal，fast。"

fn:在动画完成时执行的函数，每个元素执行一次。

#### ****[speed], [easing ], [fn ]****String,String,FunctionV1.4.3

speed: 隐藏/显示 效果的速度。默认是 "0"毫秒。可能的值：slow，normal，fast。"

easing:(Optional) 用来指定切换效果，默认是"swing"，可用参数"linear"

fn:在动画完成时执行的函数，每个元素执行一次。

#### **switch**BooleanV1.3

用于确定显示/隐藏的开关。如：true - 显示元素，false - 隐藏元素

### 示例

#### 无参数描述:

对表格切换显示/隐藏

##### jQuery 代码:

$('td).toggle();

#### fn,fn2描述:

对表格的切换一个类

##### jQuery 代码:

$("td").toggle(

function () {

$(this).addClass("selected");

},

function () {

$(this).removeClass("selected");

}

);

#### speed 描述:

用600毫秒的时间将段落缓慢的切换显示状态

##### jQuery 代码:

$("p").toggle("slow");

#### speed,fn 描述:

用200毫秒将段落迅速切换显示状态，之后弹出一个对话框。

##### jQuery 代码:

$("p").toggle("fast",function(){

alert("Animation Done.");

});

#### switch参数描述:

如果这个参数为true ，那么匹配的元素将显示;如果false ，元素是隐藏的

##### jQuery 代码:

$('#foo').toggle(showOrHide);

//相当于

if (showOrHide) {

$('#foo').show();

} else {

$('#foo').hide();

}

## **blur([[data],fn])**

### 概述

触发每一个匹配元素的blur事件。

这个函数会调用执行绑定到blur事件的所有函数，包括浏览器的默认行为。可以通过返回false来防止触发浏览器的默认行为。blur事件会在元素失去焦点的时候触发，既可以是鼠标行为，也可以是按tab键离开的

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的blur事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:blur([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的blur事件中绑定的处理函数。

### 示例

#### 描述:

触发所有段落的blur事件

##### jQuery 代码:

$("p").blur();

#### 描述:

任何段落失去焦点时弹出一个 "Hello World!"在每一个匹配元素的blur事件中绑定的处理函数。

##### jQuery 代码:

$("p").blur( function () { alert("Hello World!"); } );

## **change([[data],fn])**

### 概述

触发每个匹配元素的change事件

这个函数会调用执行绑定到change事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回false来防止触发浏览器的默认行为。change事件会在元素失去焦点的时候触发，也会当其值在获得焦点后改变时触发。

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的change事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:change([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的change事件中绑定的处理函数。

### 示例

#### 描述:

触发被选元素的 change 事件。

##### jQuery 代码:

$(selector).change();

#### 描述:

给所有的文本框增加输入验证

##### jQuery 代码:

$("input[type='text']").change( function() {

// 这里可以写些验证代码

});

## **click([[data],fn])**

### 概述

触发每一个匹配元素的click事件。

这个函数会调用执行绑定到click事件的所有函数。

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的click事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:click([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的click事件中绑定的处理函数。

### 示例

#### 描述:

触发页面内所有段落的点击事件

##### jQuery 代码:

$("p").click();

#### 描述:

将页面内所有段落点击后隐藏。

##### jQuery 代码:

$("p").click( function () { $(this).hide(); });

## **dblclick([[data],fn])**

### 概述

触发每一个匹配元素的dblclick事件。

这个函数会调用执行绑定到dblclick事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回false来防止触发浏览器的默认行为。dblclick事件会在元素的同一点双击时触发。

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的dblclick事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:dblclick([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的dblclick事件中绑定的处理函数。

### 示例

#### 描述:

给页面上每个段落的双击事件绑上 "Hello World!" 警告框

##### jQuery 代码:

$("p").dblclick( function () { alert("Hello World!"); });

## **error([[data],fn])**

### 概述

触发每一个匹配元素的error事件。

这个函数会调用所有绑定到error事件上的函数，包括在对应元素上的浏览器默认行为。可以通过在某个绑定的函数中返回false来防止触发浏览器的默认行为。error事件通常可以在元素由于点击或者tab导航失去焦点时触发。

对于error事件，没有一个公众的标准。在大多数浏览器中，当页面的JavaScript发生错误时，window对象会触发error事件;当图像的src属性无效时，比如文件不存在或者图像数据错误时，也会触发图像对象的error事件。

如果异常是由window对象抛出，事件处理函数将会被传入三个参数：

1. 描述事件的信息 ("varName is not defined", "missing operator in expression", 等等.),

2. 包含错误的文档的完整URL

3. 异常发生的行数 如果事件处理函数返回true，则表示事件已经被处理，浏览器将认为没有异常。

更多相关信息:

[msdn - onerror Event](http://msdn2.microsoft.com/en-us/library/ms536930.aspx)

[Gecko DOM Reference - onerror Event](http://developer.mozilla.org/en/docs/DOM:window.onerror)

[Gecko DOM Reference - Event object](http://developer.mozilla.org/en/docs/DOM:event)

[Wikipedia: DOM Events<](http://en.wikipedia.org/wiki/DOM_Events)

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的error事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:error([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的error事件中绑定的处理函数。

### 示例

#### 描述:

在服务器端记录JavaScript错误日志:

##### jQuery 代码:

$(window).error(function(msg, url, line){

jQuery.post("js\_error\_log.php", { msg: msg, url: url, line: line });

});

#### 描述:

隐藏JavaScript错误:

##### jQuery 代码:

$(window).error(function(){

return true;

});

#### 描述:

给你IE的用户隐藏无效的图像:

##### jQuery 代码:

$("img").error(function(){

$(this).hide();

});

## **focus([[data],fn])**

### 概述

触发每一个匹配元素的focus事件。

可以通过鼠标点击或者键盘上的TAB导航触发。这将触发所有绑定的focus函数，注意，某些对象不支持focus方法。

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的focus事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:focus([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的focus事件中绑定的处理函数。

### 示例

#### 描述:

当页面加载后将 id 为 'login' 的元素设置焦点:

##### jQuery 代码:

$(document).ready(function(){

$("#login").focus();

});

#### 描述:

使人无法使用文本框:

##### jQuery 代码:

$("input[type=text]").focus(function(){

this.blur();

});

## **focusin([data],fn)**

### 概述

在每一个匹配元素的focusin事件中绑定一个处理函数。

当一个元素，或者其内部任何一个元素获得焦点的时候会触发这个事件。这跟focus事件区别在于，他可以在父元素上检测子元素获取焦点的情况。

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的focusin事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:focusin([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的focusin事件中绑定的处理函数。

### 示例

#### 描述:

获得焦点后会触发动画:

##### HTML 代码:

<p><input type="text" /> <span>focusout fire</span></p>

<p><input type="password" /> <span>focusout fire</span></p>

##### jQuery 代码:

$("p").focusin(function() {

$(this).find("span").css('display','inline').fadeOut(1000);

});

## **focusout([data],fn)**

### 概述

在每一个匹配元素的focusout事件中绑定一个处理函数。

当一个元素，或者其内部任何一个元素失去焦点的时候会触发这个事件。这跟blur事件区别在于，他可以在父元素上检测子元素失去焦点的情况。

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的focusout事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:focusout([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的focusout事件中绑定的处理函数。

### 示例

#### 描述:

失去焦点后会触发动画:

##### HTML 代码:

<p><input type="text" /> <span>focusout fire</span></p>

<p><input type="password" /> <span>focusout fire</span></p>

##### jQuery 代码:

$("p").focusout(function() {

$(this).find("span").css('display','inline').fadeOut(1000);

});

## **keydown([[data],fn])**

### 概述

触发每一个匹配元素的keydown事件

这个函数会调用执行绑定到keydown事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回false来防止触发浏览器的默认行为。keydown事件会在键盘按下时触发。

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的keydown事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:keydown([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的keydown事件中绑定的处理函数。

### 示例

#### 描述:

在页面内对键盘按键做出回应，可以使用如下代码:

##### jQuery 代码:

$(window).keydown(function(event){

switch(event.keyCode) {

// ...

// 不同的按键可以做不同的事情

// 不同的浏览器的keycode不同

// 更多详细信息: http://unixpapa.com/js/key.html

// ...

}

});

## **keypress([[data],fn])**

### 概述

触发每一个匹配元素的keypress事件

T这个函数会调用执行绑定到keydown事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回false来防止触发浏览器的默认行为。keydown事件会在键盘按下时触发

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的keypress事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:keypress([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的keypress事件中绑定的处理函数。

### 示例

#### 描述:

计算在输入域中的按键次数：

##### jQuery 代码:

$("input").keydown(function(){  
 $("span").text(i+=1);  
});

## **keyup([[data],fn])**

### 概述

触发每一个匹配元素的keyup事件

这个函数会调用执行绑定到keyup事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回false来防止触发浏览器的默认行为。keyup事件会在按键释放时触发。

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的keyup事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:keyup([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的keyup事件中绑定的处理函数。

### 示例

#### 描述:

当按下按键时，改变文本域的颜色：

##### jQuery 代码:

$("input").keyup(function(){

$("input").css("background-color","#D6D6FF");

});

## **mousedown([[data],fn])**

### 概述

在每一个匹配元素的mousedown事件中绑定一个处理函数。

mousedown事件在鼠标在元素上点击后会触发

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的mousedown事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:mousedown([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的mousedown事件中绑定的处理函数。

### 示例

#### 描述:

当按下鼠标按钮时，隐藏或显示元素：

##### jQuery 代码:

$("button").mousedown(function(){

$("p").slideToggle();

});

## **mouseenter([[data],fn])**

### 概述

当鼠标指针穿过元素时，会发生 mouseenter 事件。该事件大多数时候会与mouseleave 事件一起使用。

与 mouseover 事件不同，只有在鼠标指针穿过被选元素时，才会触发 mouseenter 事件。如果鼠标指针穿过任何子元素，同样会触发 mouseover 事件。

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的mouseenter事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:mouseenter([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的mouseenter事件中绑定的处理函数。

### 示例

#### 描述:

当鼠标指针进入（穿过）元素时，改变元素的背景色：

##### jQuery 代码:

$("p").mouseenter(function(){

$("p").css("background-color","yellow");

});

## **mouseleave([[data],fn])**

### 概述

当鼠标指针离开元素时，会发生 mouseleave 事件。该事件大多数时候会与mouseenter 事件一起使用。

与 mouseout 事件不同，只有在鼠标指针离开被选元素时，才会触发 mouseleave 事件。如果鼠标指针离开任何子元素，同样会触发 mouseout 事件。

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的mouseleave事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:mouseleave([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的mouseleave事件中绑定的处理函数。

### 示例

#### 描述:

当鼠标指针离开元素时，改变元素的背景色：：

##### jQuery 代码:

$("p").mouseleave(function(){

$("p").css("background-color","#E9E9E4");

});

## **mousemove([[data],fn])**

### 概述

在每一个匹配元素的mousemove事件中绑定一个处理函数。

mousemove 事件通过鼠标在元素上移动来触发。事件处理函数会被传递一个变量——事件对象，其.clientX 和 .clientY 属性代表鼠标的坐标

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的mousemove事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:mousemove([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的mousemove事件中绑定的处理函数。

### 示例

#### 描述:

获得鼠标指针在页面中的位置：

##### jQuery 代码:

$(document).mousemove(function(e){

$("span").text(e.pageX + ", " + e.pageY);

});

## **mouseout([[data],fn])**

### 概述

在每一个匹配元素的mouseout事件中绑定一个处理函数。

mouseout事件在鼠标从元素上离开后会触发

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的mouseout事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:mouseout([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的mouseout事件中绑定的处理函数。

### 示例

#### 描述:

当鼠标从元素上移开时，改变元素的背景色：

##### jQuery 代码:

$("p").mouseout(function(){

$("p").css("background-color","#E9E9E4");

});

## **mouseover([[data],fn])**

### 概述

在每一个匹配元素的mouseover事件中绑定一个处理函数。

mouseover事件会在鼠标移入对象时触发

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的mouseover事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:mouseover([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的mouseover事件中绑定的处理函数。

### 示例

#### 描述:

当鼠标指针位于元素上方时时，改变元素的背景色：

##### jQuery 代码:

$("p").mouseover(function(){

$("p").css("background-color","yellow");

});

## **mouseup([[data],fn])**

### 概述

在每一个匹配元素的mouseup事件中绑定一个处理函数。

mouseup事件会在鼠标点击对象释放时

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的mouseup事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:mouseup([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的mouseup事件中绑定的处理函数。

### 示例

#### 描述:

当松开鼠标按钮时，隐藏或显示元素：

##### jQuery 代码:

$("button").mouseup(function(){

$("p").slideToggle();

});

## **resize([[data],fn])**

### 概述

在每一个匹配元素的resize事件中绑定一个处理函数。

当文档窗口改变大小时触发

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的resize事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:resize([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的resize事件中绑定的处理函数。

### 示例

#### 描述:

让人每次改变页面窗口的大小时很郁闷的方法:

##### jQuery 代码:

$(window).resize(function(){

alert("Stop it!");

});

#### 描述:

对浏览器窗口调整大小进行计数：

##### jQuery 代码:

$(window).resize(function() {

$('span').text(x+=1);

});

## **scroll([[data],fn])**

### 概述

在每一个匹配元素的scroll事件中绑定一个处理函数。

当滚动条发生变化时触发

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的scroll事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:scroll([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的scroll事件中绑定的处理函数。

### 示例

#### 描述:

当页面滚动条变化时，执行的函数:

##### jQuery 代码:

$(window).scroll( function() { /\* ...do something... \*/ } );

#### 描述:

对元素滚动的次数进行计数：

##### jQuery 代码:

$("div").scroll(function() {

$("span").text(x+=1);

});

## **select([[data],fn])**

### 概述

触发每一个匹配元素的select事件

这个函数会调用执行绑定到select事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回false来防止触发浏览器的默认行为。

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的select事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:select([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的select事件中绑定的处理函数。

### 示例

#### 描述:

触发所有input元素的select事件:

##### jQuery 代码:

$("input").select();

#### 描述:

当文本框中文本被选中时执行的函数:

##### jQuery 代码:

$(":text").select( function () { /\* ...do something... \*/ } );

## **submit([[data],fn])**

### 概述

触发每一个匹配元素的submit事件。

这个函数会调用执行绑定到submit事件的所有函数，包括浏览器的默认行为。可以通过在某个绑定的函数中返回false来防止触发浏览器的默认行为。

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的submit事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:submit([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的submit事件中绑定的处理函数。

### 示例

#### 描述:

提交本页的第一个表单:

##### jQuery 代码:

$("form:first").submit();

#### 描述:

如果你要阻止表单提交:

##### jQuery 代码:

$("form").submit( function () {

return false;

} );

## **unload([[data],fn])**

### 概述

在每一个匹配元素的unload事件中绑定一个处理函数。

### 参数

#### ****fn****FunctionV1.0

在每一个匹配元素的unload事件中绑定的处理函数。

#### ****[data],fn****String,FunctionV1.4.3

****data****:unload([Data], fn) 可传入data供函数fn处理。

****fn****:在每一个匹配元素的unload事件中绑定的处理函数。

### 示例

#### 描述:

页面卸载的时候弹出一个警告框:

##### jQuery 代码:

$(window).unload( function () { alert("Bye now!"); } );