44 第三十章 jQuery 事件 event对象遍历（三）

jQuery [2]

一、事件操作

1、 `hover( )` 滑入滑出事件

> hover(**fn1**,**fn2**)—它包含滑入滑出事件
>
> - `fn1` 函数代表滑入事件
> - `fn2` 函数代表滑出事件
>
> 只写一个代表执行相同的函数

2、 `on(` events,[selector],[data],fn `)` 事件绑定

可以是event事件，系统 事件，表单事件,还可以**自定义事件**
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。

- `绑定单个事件` ，

```
1. $('.box').on('click',function(){})
2. $('.box').on('abc',function(){})//自定义abc事件，怎么触发后面讲
```

- `绑定多个事件` ，每个事件执行相同的内容

```
1. $('.box').on('click hover',function(){})
```

- `绑定多个事件` ，每个事件执行不同的内容

```
$('.box').on({
'click': function(){}
'hover': function(){}
})
```

- `事件委托` ，每个事件执行不同的内容

```
$('#div1').on('click','li',function(){
    alert('ok');
})
$('#div1').append($('<li>new</li>'));
```

on()方法 提供绑定事件处理程序所需的所有功能,替换旧的see .bind(), .delegate()

3、 `one(` type,fn `)`

> 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**

4、 `off()` 取消事件

- ele.`off()` 取消对象所有事件
- ele.`off(` ‘click’ `)` 取消对象指定事件

```
$('.box').on('click hover',function(){
                alert(123);
            $('.box').off('click');

})
```

`delegate(` selector,[type],[data],fn `)` 事件委托

> jQuery 3.0中已弃用此方法，请用 `on()` 代替。
> 使用 delegate() 方法的事件处理程序适用于当前或未来的元素（比如由脚本创建的新元素）。

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

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

`undelegate()` 解除事件委托

> 删除由 delegate() 方法添加的一个或多个事件处理程序。
> jQuery 3.0中已弃用此方法，请用 `off()` 代替。

`trigger(` type,[data]`)` 主动触发

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

```
$("p").bind("myEvent", function (event, message1, message2) {
  alert(message1 + ' ' + message2);
});
$("p").trigger("myEvent", ["Hello","World!"]);
```

`triggerHandler(` type, [data]`)`

> 跟trigger一样，但不会执行浏览器默认动作，**也不会产生事件冒泡**。
> 这个方法的行为表现与trigger类似，但有以下三个主要区别：
>
> - 第一，**他不会触发浏览器默认事件**。
> - 第二，只触发jQuery对象集合中**第一个元素**的事件处理函数。
> - 第三，这个方法的返回的是事件处理函数的返回值

二、事件对象

event.`data`

事件传递过来的数据

```
$("a").each(function(i) {
    $(this).bind('click', {index:i}, function(e){
       alert('my index is ' + e.data.index);
    });
  });
```

event.`target` 事件源

最初触发事件的DOM元素。
这是注册事件时的对象，或者它的子元素。通常用于比较 event.target 和 this 来确定事件是不是由于冒泡而触发的。经常用于事件冒泡时处理事件委托。

event.`type` 事件类型

event.`pageX`

鼠标相对于文档的左边缘的位置。

event.`pageY`

鼠标相对于文档的上边缘的位置。

event.`which`

针对键盘和鼠标事件，这个属性能确定你到底按的是哪个键或按钮

```
  $('#whichkey').bind('keydown',function(e){
  $('#log').html(e.type + ': ' +  e.which );  });
```

event.`preventDefault()` 阻止默认事件

event.`stopPropagation()` 阻止冒泡事件

> 这个方法对 trigger() 来自定义的事件同样有效

二、遍历

1、`each( fn )`

> $(selector).each(function(index,element))为每个匹配元素规定运行的函数
> index - 当前元素的 index 索引
> element - 当前的元素（也可使用“`this`”选择器）

```
1.  $("li").each(function(index,ele){
2.      console.log(index);
3.      console.log(ele);
4.   });
```

2、`map( fn )`

> $(selector).map(function(index,element))为每个匹配元素规定运行的函数
> **index** - 当前元素的 index 索引
> **element** - 当前的元素（也可使用“`this`”选择器）
> map跟each很像，不同的是map返回的是jQuery 封装的数组

```
1. $("p").append( $("input").map(function(){
2.   return $(this).val();
3. }).get().join(", ") );
```