# 一 jQuery事件

## 1 jQuery事件机制

jQuery的事件机制，指的是：jQuery对JavaScript操作DOM事件的封装，包括了：事件绑定、事件解绑、事件触发。

## 2 jQuery事件的发展

简单事件绑定 >> bind事件绑定 >> delegate事件绑定 >> on【重点】

* 简单事件绑定：

click(handler) 单击事件

blur(handler) 失去焦点事件

mouseenter(handler) 鼠标进入事件

mouseleave(handler) 鼠标离开事件

dbclick(handler) 双击事件

change(handler) 改变事件，如：文本框值改变，下来列表值改变等

focus(handler) 获得焦点事件

keydown(handler) 键盘按下事件

* bind方式 不推荐，1.7以后的jQuery版本被on取代

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

//事件响应方法

});

// 绑定单击事件处理程序，参数1：事件类型；参数2：事件处理程序

比简单事件绑定方式的优势：可以同时绑定多个事件，比如：bind(“mouseenter mouseleave”, function(){})，其缺点是：要绑定事件的元素必须存在文档中。

* delegate方式 性能高（减少绑定次数），支持动态创建的元素绑定事件

$(".parentBox").delegate("p", "click", function(){

//为 .parentBox下面的所有的p标签绑定事件

});

//参数1：要绑定事件的元素，参数2：事件类型，参数3：事件处理函数

## 3 on方式绑定事件

jQuery**1.7版本后**，jQuery用on统一了所有的事件处理的方法，on方式绑定事件是最现代的方式，兼容zepto。

语法：

$(selector).on(events[,selector][,data],handler);

//参数1：events，绑定事件的名称，可以是由空格分隔的多个事件

//参数2：selector, 执行事件的后代元素

//参数3：data，传递给处理函数的数据，事件触发的时候通过event.data来使用

//参数4：handler，事件处理函数

$(selector).on( "click",“span”, function() {});

//给$(selector)绑定事件，必须是它的内部元素span才能执行这个事件

$(selector).on(“click mouseenter”, function(){});

//绑定多个事件:表示给$(selector)匹配的元素绑定单击和鼠标进入事件

//绑定一个方法

$( "#dataTable tbody tr" ).on( "click", function() {

console.log( $( this ).text() );

});

//给子元素绑定事件

$( "#dataTable tbody" ).on( "click", "tr", function() {

console.log( $( this ).text() );

});

//绑定多个事件的方式

$( "div.test" ).on({

click: function() {

$( this ).toggleClass( "active" );

}, mouseenter: function() {

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

}, mouseleave: function() {

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

}

});

## 4 one绑定一次事件

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

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

});

## 5 事件解绑

* unbind() 方式 解绑 bind方式绑定的事件

$(selector).unbind(); //解绑所有的事件

$(selector).unbind(“click”); //解绑指定的事件

* undelegate() 方式 解绑delegate方式绑定的事件

$( selector ).undelegate(); //解绑所有的delegate事件

$( selector).undelegate( “click” ); //解绑所有的click事件

* **offf()方式** **解绑on方式绑定的事件**

// 解绑匹配元素的所有事件

$(selector).off();

// 解绑匹配元素的所有click事件

$(selector).off(“click”);

// 解绑所有代理的click事件，元素本身的事件不会被解绑

$(selector).off( “click”, “\*\*” );

案例1：

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 );

案例2：（了解）解绑命名空间的方式：

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" );

## 6 事件触发

$(selector).click(); //简单事件触发：触发 click事件

$(selector).trigger(“click”); //trigger方法触发事件

$(selector).triggerHandler(“focus”); //此方式不触发浏览器行为如文本框获焦点

## 7 jQuery事件对象介绍

event.data 传递给事件处理程序的额外数据

event.currentTarget 等同于this，当前DOM对象

event.pageX  鼠标相对于文档左部边缘的位置

**event.target** 触发事件源，不一定===this

**event.stopPropagation()；** 阻止事件冒泡

**event.preventDefault();**  阻止默认行为

event.type 事件类型：click，dbclick…

event.which 鼠标的按键类型：左1 中2 右3

event.keyCode 键盘按键代码

## 8 阻止冒泡

event.stopPropagation() //阻止事件冒泡

event.preventDefault();  //阻止默认行为

# 二 $下的常用方法

$下的方法大多数为工具类方法，这些方法不仅可以给jQuery使用，也可以给原生JS使用。$也可以书写为jQuery。

## 1 type()

判断类型，与typeof区别是可以判断对象类型，比如时间对象返回Date，而typeof返回的都是Object。

## 2 trim()

去除空白；

## 3 inArray()

类似indexOf();

    var arr = ['a','b','c','d'];

    alert( $.inArray('b',arr) );

## 4 proxy()

改变this指向。

$(function(){

    function show(n1,n2){

        alert(n1);

        alert(n2);

        alert(this);

    }

    //show();

    //$.proxy(show , document,3)(4);

    $(document).click( $.proxy(show,window,3,4) );

});

## 5 parseJSON()

将字符串数据转换成json对象

var str = '{"name":"hello"}';

alert($.parseJSON( str ).name);

## 6 makeArray()

将类数组转换成真正的数组。

    var aDiv = document.getElementsByTagName('div'); //类数组

    $.makeArray(aDiv).push();

## 7 map函数

$.map(arry,function(object,index){}) 返回一个新的数组

$("li").map(function(index, element){}) 注意参数的顺序是反的

var newArr = $.map($("li"), function(i, e) {

return $(e).text() + i;//每一项返回的结果组成新数组

});

var newArr = $("li").map(function(elem, index){

console.log("elem:" + elem);

console.log("index:" + index);

retrun index;

});

# 三 jQuery补充

## 1 链式编程

链式编程原理：return this;

通常情况下，只有设置操作才能把链式编程延续下去。因为获取操作的时候，会返回获取到的相应的值，无法返回 this。

end(); // 结束当前链最近的一次过滤操作，并且返回匹配元素之前的状态。

## 2 隐式迭代

在方法的内部会为匹配到的所有元素进行循环遍历，执行相应的方法；而不用我们再进行循环，简化我们的操作，方便我们调用。

如果获取的是多元素的值，大部分情况下返回的是第一个元素的值。

## 3 each方法

有了隐式迭代，为什么还要使用each函数遍历？

大部分情况下是不需要使用each方法的，因为jQuery的隐式迭代特性。

如果要对每个元素做不同的处理，这时候就用到了each方法

作用：遍历jQuery对象集合，为每个匹配的元素执行一个函数

// 参数一表示当前元素在所有匹配元素中的索引号

// 参数二表示当前元素（DOM对象）

$(selector).each(function(index,element){});

$( "li" ).each(function() {

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

});

$( "li" ).each(function( index ) {

console.log( index + ": " + $( this ).text() );

});

$( "div" ).each(function( index, element ) {});

## 4 多库共存

此处多库共存指的是：jQuery占用了$ 和jQuery这两个变量。当在同一个页面中引用了jQuery这个js库，并且引用的其他库（或者其他版本的jQuery库）中也用到了$或者jQuery这两个变量，那么，要保证每个库都能正常使用，这时候就有了多库共存的问题。

// 模拟另外的库使用了 $ 这个变量

// 此时，就与jQuery库产生了冲突

var $ = { name : “itecast” };

解决方式：

// 作用：让jQuery释放对$的控制权，让其他库能够使用$符号，达到多库共存的目的。此后，只能使用jQuery来调用jQuery提供的方法

$.noConflict();

var myQuery = $.noConflict();

var $ = 10;

myQuery(function(){

    myQuery('body').css('background','red');

});

## 4 jQuery插件机制

jQuery这个js库，虽然功能强大，但也不是面面俱到包含所有的功能，通过插件的方式，可以扩展它的功能。

$.extend: 扩展工具方法下的插件形式

$.fn.extend: 扩展到JQ对象下的插件形式

<script>

$.extend({

    leftTrim : function(str){

        return str.replace(/^\s+/,'');

    },

    rightTrim : function(){

        //方法体

    }

});

</script>

<script>

var str = ' hello ';

alert( '('+$.leftTrim(str)+')' );

</script>

<script>

$.fn.extend({

    aaa : function(){

        alert(2);

    }

});

</script>

<script>

$().aaa();

</script>

全局jQuery函数扩展方法

$.pluginName = function() {};

jQuery对象扩展方法

$.fn. pluginName = function() {};

(function ($) {

$.fn.lxCfbg = function (options) {

// 合并参数

var opts = $.extend({

"color": "#000",

"font-size": "16px",

"background-color": "#fff"

}, options);

$this.css({

"color": opts.color,

"font-size": opts["font-size"],

"background-color": opts["background-color"]

});

return $this;

};

})(jQuery);

# 四 jQueryUI

jQueryUI专指由jQuery官方维护的UI方向的插件。

官方API：<http://api.jqueryui.com/category/all/>

其他教程：http://www.runoob.com/jqueryui/jqueryui-intro.html

基本使用:

1. 引入jQueryUI的样式文件
2. 引入jQuery
3. 引入jQueryUI的js文件
4. 使用jQueryUI功能

继续深入的话，我们还应该掌握哪些？

$.Callbacks() : 回调对象

deferred() : 延迟对象

$.hodeReady() : 持有和释放ready

$.dequeue() : 执行队列

$.support : 功能检测