*[jQuery](#_Toc17678)* [1](#_Toc17678)

[jQuery选择器 2](#_Toc6694)

[增删改 2](#_Toc26945)

[CSS & attr 3](#_Toc10363)

[jQuery&ajax 3](#_Toc24095)

[jQuery遍历（traverse） && 效果 5](#_Toc14442)

[jQuery事件 6](#_Toc30807)

# ***jQuery***

jquery是一个JavaScript库，一个功能强劲的工具。有了它，你可以少写很多JavaScript代码。不仅是效率提升器，更是懒癌患者的福音。它的运行原理，简单来讲就是选中某元素，然后对该元素进行操作。

$(“#wg”).shower（）; // 这个是我编的

看清了，俩括号。我原本以为选中某元素后还可以像原始js那样通过属性（.value）访问。结果当然是不行啦！用jQuery选中元素后只能跟要调用的函数。

上面那个只是jQuery选择器的一种，也是最常用的一种。jQuery选择器是基于css的，通过id，类，类型，属性值等等来查找。

刚接触jQuery时很不适应它的写法，像这样。

$(document).ready(function(){

$(“#myButton”).click(function(){

...

});

});

回调回调回调，没完没了。给触发事件“赋”函数，像上面例子的意思就是在文档加载完毕后给某按钮增加onclick事件。用纯JavaScript肯定也能实现，但这个，我还真没研究过。

哦，还有，$(document).ready();这个方法可以重复调用的，我之前也试过，但。。。我是多缺乏安全感。被调用顺序安写的顺序来（代码总是从头执行到尾的）。

## jQuery选择器

这个东西的门道还蛮多的，就目前而言我只用过id选择器、类选择器、标签选择器。一般用用的话，这三个差不多了。再往下无非两个概念，交集（一块写）与并集（加逗号）。然后我们明确一条规则，不管你要怎么写选择器，内容统统放在“ ”内！

[href=‘#’] 这是一个中括号，用来匹配属性，左边的例子会匹配所有href属性为#的dom节点。那个等于可以不写，又或者换成 != $=（以什么什么结尾的意思）

:checked 返回所有被选中的input元素，这个在我看来，还是一种属性，只不过写在或是隐藏在<input>的最右，像这样：

<input type=”radio”name=”myRadio”checked>

## 增删改

***增***

$(“<div></div>”)

// 创建dom节点

***删***

empty()&remove()

// 清空子节点与删除节点（包括子节点）

***改***

before()&after()

insertBefore()&insertAfter()

这四个函数头一回碰到还是有点绕的。到底谁在前，谁在后。jQuery的操作基本都是以“取节点，对节点执行操作”为核心的。

<span id=*"s1"* class=*"label label-danger"*>Danger</span>

<span id=*"s2"* class=*"label label-danger"*>Success</span>

这里有两个bootstrap徽章。

$(“#s1”).after($(“#s2”)); 的意思是，取到s1，s1的后面是s2。

$(“#s1”).insertAfter($(“#s2”)); 的意思是，取s1，s1插在s2后面。

## CSS & attr

$(“div”).addClass(“panel panle-default”);

// 增加类

$(“div”).attr(“id”);

// 这样会返回id

相应的，$(“div”).attr(“id”, “123”);这样可以设置id。另外jQuery可以批量设置属性 ，简单来讲就是将单个var参数，换成JavaScript对象。

1.6版本的jQuery新增prop() 函数，它是property的缩写，attr是attribute的缩写，二者都是属性的意思，但操作对象其实不一样。

// 老实讲这个“不一样”，我觉得蛮有限的

prop()针对HTML元素固有的属性，attr()针对明确给出的属性。举个例子：

<input type=”radio”name=”myRadio”>

则 $(“[name=’myRadio’]”).prop.(“checked”)返回false

而 $(“[name=’myRadio’]”).attr.(“checked”)返回undefined

看了一篇博客，元素自带的属性推荐用prop()，自定义属性用attr()。

## jQuery&ajax

1.

var jqxhr =

$.ajax({

// 请求类型

type: "get",

// 请求的URL

url: "/theServiceToCall.html",

// 加了这个可以将服务器传来的json数据转换为对象

dataType: 'json',

// 传往服务器的数据

data: {

name : "The name",

desc : "The description"

},

// 要异步还是同步

async: false,

// 要不要读浏览器缓存

cache: false

}).done(function(data, textStatus, jqXHR)

{

alert("Success: " + response);

}).fail(function(jqXHR, textStatus, errorThrown)

{

alert("Error");

}).always(function(jqXHROrData, textStatus, jqXHROrErrorThrown)

{

alert("complete");

});

// 这段代码其实不难看懂，但很揪心，截止至目前为止最让我揪心的一段代码。jQuery ajax 果然没这么简单啊。

呐，这个呢，$.ajax()肯定会返回一个对象，然后调用了该对象的done()函数，然后done()继续返回对象在调用fail()轮回轮回。但done()只有ajax请求成功时才会调用，fail()失败才会调用，always()无论如何都会调用。

无论done()、fail()还是always()都有一个回调函数，不过参数都差不多。

data是服务器返回的数据。

textStatus是服务器返回的状态（success/？）学名文本状态信息，有逼格。

jqXHR是$.ajax(）函数返回的，然后我就不懂了，怎么获取到这个参数的，难道说他不需要像Java那样显示的通过引用传值？jQueryXMLHttpRequest的缩写？

errorThrown是jQuery抛出的error。

$.ajax()这个方法远比我给的这个例子复杂。但东西太多，一时半会又用不到，所以这里先写个目录一样的东西。

发送数据是否“原生”

发送方式为get/post

2. load（）

$(selector).load(URL,data,callback);

从服务器加载数据，并将数据放入选中的元素里。（我已开始还奇怪为什么这玩意能将数据直接显示在网页里。唉，naive啊。）

URL是必须的，请求地址。

data 是可选的，json格式键值对。不过嘿，就不能直接写在url后面？反正效果都一样。

callback是回调函数，肯定会调用，不管后台有没有加载成功。

3. Get/Post

* $.post(URL,data,callback);
* $.get(URL,callback);

上面这两个方法其实是阉割版的 $.ajax() ，用于发送简化版的HTTP GET/POST请求。

var parameters = { p1 : "val1", p2 : "val2"};

$.post("data.html", parameters )

.done(function(data) {

$("#targetElement").html(data);

}) ;

$.get() & $.post()会返回jqXHR对象，因此处理他的方式和$.ajax()差不多。不过怎么说呢， .done() 又或者回调看个人喜好吧，我不是很喜欢回调，done（）方法用的应该多一点。

## jQuery遍历（traverse） && 效果

1. each()

//先介绍一个很有趣的

$(“.column”).each(function(index, element){

....

});

这个方法会遍历每一个被选中元素，并用回调函数进行操作。这里要留心回调函数的两个参数，index表示被选中元素的序号，element表示当前正在遍历的元素，注意是元素也就是说它是DOM节点因此也只有DOM节点的方法可以用。不过在外面套层$(element)就可以愉快的使用jQuery了。

1. parent()

返回上一级元素（祖先）

// 这两个东西配合上window.event.srcElement就是神器！

1. prev() && next()

上一个同胞与下一个同胞

4. fadeOut()

$(“#sb”).fadeOut(“slow”,function(){

...

}); // 回调方法用不用随意，slow参数用不用随意，一个参数都不要亦可

## jQuery事件

* 给事件添加回调函数是件很有意思的事

$(“input”).click(function(){ // 这是最常用的方法

....

});

function myFunc(){...}

$(“input”).click(myFunc); // 这个用的相对少一点

那么问题来了，如果要添加的回调函数是需要参数的，要怎么办呢？得包起来

function myFunc(data){ ... }

$(“input”).click(function(){

var data = ... ;

myFunc(data);

});

该部分内容不仅仅是jQuery，页面JavaScript同理。也很有可能这就是从JavaScript那继承来的写法。

* submit()事件

仅对<form>元素有效果。而且有意思的是，该函数不仅仅是提交，还可以添加事件。

例：

$(“#form”).submit(function(){

alert(“提交的时候闪一下。”);

});

新鲜哎嘿~

* toggle()事件

toggle的中文意思有挂索桩，套索钉。Bootstrap模态框里有个属性是data-toggle数据牵挂处？瞎猜的。

$(document).ready(function(){

$(“#xxx”).toggle(

function(){

alert(“第一次”);

},

function(){

alert(“第二次”);

});

});

可以定义多个函数，随着点击不停切换。

* $("form").serialize()

######################## jQuery.blockUI.js ########################

http://bookshadow.com/weblog/2014/09/26/jquery-blockui-js-introduction/

######################## Mustache #############################

// 不光光是js插件

模板引擎，刚看到这个概念时还糊涂了一下，然后才反应过来，jsp不就是模板引擎么。而这个mustache就是为前端服务的，比如将后台传来的json数据渲染成HTML代码。因此学mustache好比学jsp，并且要简单的多，因为他没有那么多杂七杂八的标签、对象。

***Mustache的标签***

只要定义好一个模板，并为该模板提供数据源，剩下的程序统统帮你搞定。为模板填充数据时就需要用到标签，因为不同的数据要填充到不同的地方。为了与HTML的 < > 区分开，mustache用 {{ }} 做为标签。

* {{prop}}

使用该标签后，数据源中的数据会以字符串的形式填充至模板中名为“prop”的标签处。

* {{{prop}}}
* {{#prop}}{{/prop}}
* {{^prop}}{{/prop}}

############################ z-tree ###########################

开始进行设置

**var** setting = {

view: {

addHoverDom: addHoverDom,

removeHoverDom: removeHoverDom,

selectedMulti: **false**

/\*

addHoverDom与removeHoverDom都是函数。当鼠标移动到节点上时，在节点后显示一个按钮或删除一个按钮。不写这两个属性，默认也会显示修改按钮与删除按钮，不知道是不是demo的问题。

\*/

},

callback : {

// 禁止拖拽

beforeDrag : **function**(){**return** **false**;},

// 子节点的点击事件

onClick: childrenOnClick

},

check: {

enable: **true**

},

data: {

simpleData: {

enable: **true**

}

},

edit: {

enable: **true**

}

};

############################# 一些例子 ########################

* 鼠标移入，边框与内容同色，鼠标移除，复原。Footer始终置于底部。

$(document).ready(function(){

$(".content-body").on("mouseover",".portal-item:not('.inactive')",function(e) {

var color = $(this).find(".icon-wrapper").css("background-color");

$(this).css("border-color",color);

$(this).find("h4,p").css("color",color);

});

$(".content-body").on("mouseout",".portal-item",function(e) {

$(this).css("border-color","transparent");

$(this).find("h4").css("color","#333");

$(this).find("p").css("color","#777");

});

var ph = $(window).height() - getHeight("#wrapper-main") - getHeight("#footer");

if(ph > 0) {

$("#footer").css('margin-top',ph + "px")

}

$(window).resize(function() {

var ph = $(window).height() - getHeight("#wrapper-main") - getHeight("#footer");

if(ph > 0) {

$("#footer").css('margin-top',ph + "px")

}

});

});

// 有空慢慢研究，现在先复制粘贴