jQuery1.7.1 API 手册

本文基于 jQuery1.7.1 版本，是对官方 API 的整理和总结，完整的官方 API 见 http://api.jquery.com/browser/

0、总述

jQuery 框架提供了很多方法，但大致上可以分为 3 大类：获取 jQuery 对象的方法、在 jQuery 对象间跳转的方法，以及获取 jQuery 对象后调用的方法

其中第一步是怎样获取 jQuery 对象。大致来说，是通过最核心的$()方法，将页面上的元素（或者在页面上不存在的 html 片段）包装成 jQuery 对象。

$()方法里面支持的语法又包括 3 大类，分别是表达式（包括类表达式.，id 表达式#，元素表达式等）、符号（包括后代符号 space，next 符号+等）、过滤器（包括:过滤器和[]过滤器）。

通过以上 3 种的组合，“查询”得到想要操作的元素或者元素集合，作为$()的参数，得到 jQuery 对象（或者 jQuery 对象的集合）

第二步是在 jQuery 对象间的跳转。也就是说，已经得到了一个 jQuery 对象，但是并不是想要的，那么可以通过一系列的跳转方法，比如 parent()、next()、children()、find()等，或者过滤筛选的方法，比如 eq()、filter()、not()等，来得到最终想要操作的 jQuery 对象。

用跳转和过滤方式得到的 jQuery 结果，往往通过比较复杂的表达式组合，可以达到同样的目的。

比如说$("div").eq(3)，也可以用$("div:eq(3)")达到同样的目的。

又比如说$("div").find("span")，可以用$("div span")取到同样的元素。

方法是很灵活的，要根据具体的情况来选择。一般来说，HTML 页面写得越规范，使用 jQuery 就越简单

还有一种情况，在得到了 jQuery()对象之后，想要判断其是否满足条件，那么可以调用 is()、hasClass()等方法，返回一个 boolean 值，进行后续的判断。这类方法也可以归到这类。

第三步是在获取准确的 jQuery 对象之后，调用其上的各种方法，来进行操作。这一步反而是比较简单的了。

后面就是对 jQuery 框架各种方法的简要介绍，更详细的内容，还是以官方 API 为准

1、$(...)

$() 一切的核心，可以跟 4 种参数

$(expression)，比如$("#id")、$(".class")等，返回 jQuery 对象，或者 jQuery 对象的集合
$(html)，比如$("hello world")，返回 jQuery 对象，或者 jQuery 对象的集合
$(element)，比如$(document.body)，返回 jQuery 对象，或者 jQuery 对象的集合
$(*)，所有元素

2、jQuery Object Accessors

jQuery.index(element)，返回该 jQuery 对象在集合中的索引

jQuery.each(function)，遍历 jQuery 对象集合，在每个对象上执行 function 函数，function callback(index, domElement){this};

jQuery.size()，返回 jQuery 对象集合的大小

jQuery.length，相当于 size()方法

jQuery.get()，获取原生 DomElement 对象的 Array

jQuery.get(index)，获取原生 DomElement 对象

jQuery.eq(position)，获取 jQuery 对象集合中的一个 jQuery 对象

3、Data 相关方法

jQuery.data(name)
jQuery.data(name, value)
jQuery.removeData(name)

4、选择符

multiple(selector1, selector2)，可以选择多个元素或者表达式，包装成 jQuery 对象的集合
例子：$("div,span")

id(id)
例子：$("#id")

class(class)
例子：$(".class")

element(element)
例子：$("div")

all
例子：$("*")

descendant
例子：$("table tr td")

child(parent, child)
例子：$("#id > span")，和上一个 descendant 的区别在于，descendant 只要是后代就会被选中，而 child 必须是直接子节点，不包括孙子节点

next(prev, next)
例子：$("label + input")，选中的是 label 标签的下一个 input 标签，返回 jQuery 对象的集合

siblings(prev, siblings)
例子：$("#prev ~ div")，选中的是#prev 之后的所有 div 标签，返回 jQuery 对象的集合，有点像 next，但是范围更大

Basic Filters
$(":header")，选中所有 header，包括<h1><h2>等
$("tr:odd")，选中所有奇数行
$("tr:even")，选中所有偶数行
$(":animated")，选中所有当前有特效的元素，$("div:animated")，选中当前所有有特效的

$("tr:first")，选中第一行
$("tr:last")，选中最后一行
$("input:not(:checked)")，选中所有没有“checked”的 input 元素
$("td:gt(4)")，选中所有 index 是 4 之后的 td
$("td:lt(4)")，选中所有 index 是 4 之前的 td
$("td:eq(4)")，选中 index 是 4 的 td，可以用$("td").eq(4)来实现同样的效果

Content Filters
$("div:contains('John')")，选中所有包含"John"字符串的 div
$("td:empty")，选中所有内容为空的 td
$("div:has(p)")，选中包含有<p>元素的 元素，返回 jQuery 对象集合
$("td:parent")，选中所有包含子节点的元素，包括文本也可以算是子节点

Visibility Filters
$("span:hidden")，选中所有隐藏的
$("span:visible")，选中所有可见的

Attribute Filters
$("div[id]")，选中包含 id 属性的 元素
$("input[name$='letter']")，选中包含某个属性的<input>元素，这个属性名是以'letter'结尾的
$("input[name^='letter']")，选中包含某个属性的<input>元素，这个属性名是以'letter'开头的

$("input[name*='man']")，选中包含某个属性的<input>元素，这个属性的属性名里包含'man'
$("input[name='newsletter']")，选中包含一个属性的<input>元素，这个属性的名字是'newsletter'
$("input[name!='newsletter']")，选中所有不包含'newsletter'属性的<input>元素
$("input[id][name$='man']")，选中包含 id 属性，和以'man'结尾属性的<input>元素

Child Filters
$("ul li:nth-child(2)")，选中自身是<ul>元素的第二个子节点的<li>元素，注意这个计算是从 1 开始的，不是从 0 开始
$("div span:firstChild")，选中自身是 元素的第一个子节点的元素
$("div span:lastChild")，选中自身是 元素的最后一个子节点的元素
$("div span:onlyChild")，选中自身是 元素的唯一子节点的元素

Forms
$(":button")，所有<button>元素，和<input type="button">元素
$("form :checkbox")，选中所有<form>标签下的<input type="checkbox">，不过这样会比较慢，官方建议使用$("input:checkbox")
$(":file")，选中所有<input type="file">
$(":hidden")，选中所有隐藏元素，以及<input type="hidden">
$(":input")，选中所有<input>
$(":text")，选中所有<input type="text">
$(":password")，选中所有<input type="password">
$(":radio")，选中所有<input type="radio">，不过这样会比较慢，建议使用$("input:radio")
$(":image")，选中所有<input type="image">
$(":reset")，选中所有<input type="reset">
$(":submit")，选中所有<input type="submit">

Form Filters
$("input:enabled")，选中所有 enabled 的<input>元素
$("input:disabled")，选中所有 disabled 的<input>元素
$("input:checked")，选中所有 checked 的<input type="checkbox">元素
$("input:selected")，选中所有 selected 的<option>元素

5、属性相关的方法

jQuery.removeAttr(name)
jQuery.attr(name)，返回属性的值，比如$("img").attr("src")
jQuery.attr(key,value)，这是设置属性的值
jQuery.attr(properties)，也是设置属性的值
例子：

```
$("img").attr({
      src: "/images/hat.gif",
      title: "jQuery",
      alt: "jQuery Logo"
```

});
jQuery.attr(key,function)，也是设置属性的值，这个 function 计算出的结果，赋给 key

```
function callback(index) {
    // index == position in the jQuery object
    // this means DOM Element
}
```

6、class 相关的方法

jQuery.toggleClass(class)，反复切换 class 属性，该方法第一次执行，增加 class，然后去除该 class，循环
jQuery.toggleClass(class,switch)，增加一个 switch 表达式
jQuery.hasClass(class)，返回 boolean
jQuery.removeClass(class)，删除 class
jQueyr.addClass(class)，增加 class

7、HTML 相关的方法

jQuery.html()，返回包含的 html 文本
jQuery.html(val)，用 val 替换包含的 html 文本

8、文本相关的方法

jQuery.text()，返回包含的纯文本，不会包括 html 标签，比如abcd，调用.text()方法，只会返回 abcd，不会返回abcd
jQuery.text(val)，用 val 替换包含的纯文本，和 html(val)方法的区别在于，所有的内容会被看作是纯文本，不会作为 html 标签进行处理，比如调用.text("abcd")，和不会被认为是 html 标签

9、值相关的方法

jQuery.val()，返回 string 或者 array
jQuery.val(val)，设置 string 值
jQuery.val(array)，设置多个值，以上 3 个方法，主要都是用在表单标签里，如<input type="text">，<input type="checkbox">等

10、在 jQuery 对象集合中进行过滤

以下几类方法有点像把选择符 Filter 进行方法化，比如$("label:eq(4)")，取到第 4 个<label>元素，这个就可以用$("label").eq(4)来替代，达到同样的效果

jQuery.is(expr)，返回 boolean，比如$(this).is(":first-child")，判断一个元素，是不是其父节点的第一个子节点

jQuery.eq(index)，$("div").eq(2)，取出第 2 个 元素

jQuery.filter(expr)，比如$("div").filter(".middle")，会从 div 元素中筛选出属于 middle 的 class 的元素；再比如$("p").filter(".selected, :first")，会取出是 selected 类，或者是第一个元素的<p>元素，这个可以用$("p.class, p:first")来代替
这个方法，会从初始的结果集中，筛选保留一部分

jQuery.filter(fn)，类似于上一个函数，可以传进去一个 function，用这个 function 的返回值，进行筛选

```
function callback(index){
    // index == position in the jQuery object
    // this means DOM Element
    return boolean;
}
```

jQuery.not(expr)，是和 filter(expr)相反的方法，不是和 is(expr)相反的方法。该方法把满足 expr 的元素给排除掉，比如$("div").not(".green, #blue")，把 class 是 green 或者 id 是 blue 的元素过滤掉

jQuery.slice(start, end)，从 jQuery 对象集合中选出一段

jQuery.map(callback)，不知道是干嘛的

11、在 jQuery 对象之间查找

jQuery.parent(expr)，找父亲节点，可以传入 expr 进行过滤，比如$("span").parent()或者$("span").parent(".class")
jQuery.parents(expr)，类似于 jQuery.parent(expr)，但是是查找所有祖先元素，不限于父元素
jQuery.children(expr)，返回所有子节点，和 parents()方法不一样的是，这个方法只会返回直接的孩子节点，不会返回所有的子孙节点
jQuery.contents()，返回下面的所有内容，包括节点和文本。这个方法和 children()的区别就在于，包括空白文本，也会被作为一个 jQuery 对象返回，children()则只会返回节点

jQuery.prev()，返回上一个兄弟节点，不是所有的兄弟节点
jQuery.prevAll()，返回所有之前的兄弟节点
jQuery.next()，返回下一个兄弟节点，不是所有的兄弟节点
jQuery.nextAll()，返回所有之后的兄弟节点
jQuery.siblings()，返回兄弟姐妹节点，不分前后

jQuery.add(expr)，往既有的 jQuery 对象集合中增加新的 jQuery 对象，例子：$("div").add("p")

jQuery.find(expr)，跟 jQuery.filter(expr)完全不一样。jQuery.filter()是从初始的 jQuery 对象集

合中筛选出一部分，而 jQuery.find()的返回结果，不会有初始集合中的内容，比如$("p").find("span")，是从<p>元素开始找，等同于$("p span")

12、串联方法

jQuery.andSelf()，把最后一次查询前一次的集合，也增加到最终结果集中，比如$("div").find("p").andSelf()，这样结果集中包括所有的<p>和 。如果是$("div").find("p")，那就只有<p>，没有

jQuery.end()，把最后一次查询前一次的集合，作为最终结果集，比如$("p").find("span").end()，这样的结果集，是所有的<p>，没有

13、DOM 文档操作方法

jQuery.append(content)，这个方法用于追加内容，比如$("div").append("hello");

jQuery.appendTo(selector)，这个方法和上一个方法相反，比如$("hello").appendTo("#div")，这个方法其实还有一个隐藏的 move 作用，即原来的元素被移动了
jQuery.prepend(content)，跟 append()方法相对应，在前面插入
jQuery.prependTo(selector)，跟上一个方法相反

jQuery.after(content)，在外部插入，插入到后面，比如$("#foo").after("hello");
jQuery.insertAfter(selector)，和上一个方法相反，比如$("hello").insertAfter("#foo");
jQuery.before(content)，在外部插入，插入到前面
jQuery.insertBefore(selector)，跟上一个方法相反

jQuery.wrapInner(html)，在内部插入标签，比如$("p").wrapInner("");
jQuery.wrap(html)，在外部插入标签，比如$("p").wrap(" ")，这样的话，所有的<p>都会被各自的 包裹
jQuery.wrapAll(html)，类似上一个，区别在于，所有的<p>会被同一个 包裹

jQuery.replaceWith(content)，比如$(this).replaceWith(" "+$(this).text()+" ");
jQuery.replaceAll(selector)，比如$(" hello ").replaceAll("p");

jQuery.empty()，比如$("p").empty()，这样的话，会把<p>下面的所有子节点清空
jQuery.remove(expr)，比如$("p").remove()，这样的话，会把所有<p>移除，可以用表达式做参数，进行过滤

jQuery.clone()，复制一个页面元素

14、CSS 相关方法

jQuery.css(name)，获取一个 css 属性的值，比如$("p").css("color")
jQuery.css(object)，设置 css 属性的值，比如$("p").css({"color":"red","border":"1px red solid"});

jQuery.css(name,value)，设置 css 属性的值，比如$("p").css("color","red");

15、位置计算相关方法

jQuery.scrollLeft()，设置滚动条偏移，这个方法对可见元素或不可见元素都生效
jQuery.scrollTop()，设置滚动条偏移，这个方法对可见元素或不可见元素都生效
jQuery.offset()，计算偏移量，返回值有 2 个属性，分别是 top 和 left
jQuery.position()，计算位置，返回值也有 2 个属性，top 和 left

16、宽度和高度计算相关方法

这组方法需要结合 CSS 的盒子模型来理解，margin 始终不参与计算

jQuery.height()，这个方法计算的是 content
jQuery.innerHeight()，这个方法计算的是 content+padding
jQuery.outerHeight()，这个方法计算的是 content+padding+border
jQuery.width();
jQuery.innerWidth();
jQuery.outerWidth();

17、页面加载完成事件

$(document).ready(function(){})，可以简写为$(function(){})

18、事件绑定方法

jQuery.bind(type,data,fn)

bind()方法可以接受 3 个参数，第 1 个是事件类型，类型是 string，可能的值有 blur, focus, load, resize, scroll, unload, beforeunload, click, dblclick, mousedown, mouseup, mousemove, mouseover, mouseout, mouseenter, mouseleave, change, select,

submit, keydown, keypress, keyup, error

第 3 个参数是当事件发生时，要执行的函数，函数原型是

```
function callback(eventObject) {
    this; // dom element
}
```

在这个方法里 return false 会阻止事件冒泡并中止默认行为，如果在这个方法里调用 eventObject.preventDefault() 则会中止默认行为，如果在这个方法里调用

eventObject.stopPropagation()则只会阻止事件冒泡

第 2 个参数是可选的，会赋值给 e.data，比如

```
function handler(event) {
    alert(event.data.foo);
}
$("p").bind("click", {foo: "bar"}, handler)
```

jQuery.one(type,data,fn)，这个方法类似于 bind()方法，区别在于只会绑定一次

jQuery.unbind(type,fn)，解除绑定

jQuery.trigger(event,data)，触发事件，要注意这个方法，同样会引起浏览器的默认行为，比如 submit

另外，这个方法如果和 bind()方法里定义的 handler 配合使用，可以更加灵活地传递参数，比如

```
$("#test").bind("click", {name : "kyfxbl"}, function(e, foo) {
    alert(e.data.name);
    alert("foo: " + foo);
});
```

以上代码，如果直接点击#test，则 foo 的值是 undefined

但是如果通过$("#test").trigger("click",["foo"])来触发，则参数 foo 会被赋值为"foo"

jQuery.triggerHandler(event,data)，这个方法和 trigger()方法十分相像，主要有 2 点不同，1 是这个方法不会触发浏览器的默认行为，2 是它只会在 jQuery 对象集合的第一个元素上触发

jQuery.live(type,fn)，这个方法十分类似 jQuery.bind()方法，区别在于这个方法对后来才添加进来的元素同样有效

jQuery.die(type,fn)，这个是 jQuery.live()的相反方法

19、事件快捷方法

jQuery.hover(over,out)，这个方法是 mouseenter 和 mouseleave 的便捷方法，2 个参数的函数原型是：

```
function callback(eventObject) {
    this; // dom element
}
```

jQuery.toggle(fn,fn2,fn3,...)，这个方法是多次点击的便捷方法，参数的函数原型是：

```
function callback(eventObject) {
    this; // dom element
}
```

jQuery 提供了两类便捷方法：

第一类是类似于 click()这种，相当于简化的 jQuery.trigger()方法，比如$("p").click()相当于$("p").trigger("click")，不过该方法，无法像完整的 jQuery.trigger("click", data)方法一样，传递一个附带的参数

第二类是类似于 click(function)这种，相当于简化的 jQuery.bind()方法，比如$("p").click(function)相当于$("p").bind("click",function)，不过该方法，无法像完整的 jQuery.bind("click", data, func)一样，传递一个额外的参数

20、切换元素显示与否的方法

jQuery.toggle()，原本显示的元素会不显示，原本不显示的会显示出来。这些元素可以是通过 show()和 hide()切换的，也可以是通过 display:none 来设置的
jQuery.show()，显示元素
jQuery.hide()，隐藏元素

jQuery.show(speed, callback)，类似于上面的 jQuery.show()，不过可以设置速度以及回调函数

speed 可以是"slow"、"normal"、"fast"，也可以是毫秒数
callback 函数的原型是：

```
function callback() {
    this; // dom element
}
```

jQuery.hide(speed, callback)
jQuery.toggle(speed, callback)

21、页面一些特效方法

jQuery.slideDown(speed, callback)，让一个元素下滑，从无到有
jQuery.slideUp(speed, callback)，让一个元素上升，从有到无
jQuery.slideToggle(speed, callback)，切换一个下滑和上升

jQuery.fadeIn(speed, callback)，淡入效果
jQuery.fadeOut(speed, callback)，淡出效果
jQuery.fadeTo(speed, opacity, callback)，变淡效果

22、ajax 相关方法

$.ajax(options)，这个是底层方法，上层的$.get()和$.post()都是基于此方法的封装

options:

async：是否异步，默认为 true
url：目标地址
type：请求类型，可以是"POST"或者"GET"
data：请求参数，比如"name=kyfxbl&location=shenzhen"

complete(function)：请求结束后的回调函数，函数原型是
function (XMLHttpRequest, textStatus) {
    this; // the options for this ajax request
}

success(function)：请求成功后的回调函数，函数原型是
function (data, textStatus) {
    // data could be xmlDoc, jsonObj, html, text, etc...
    this; // the options for this ajax request
}

例子：
```
$.ajax({
    url : "user/ajax",
    type : "GET",
    data : "name=kyfxbl&location=shenzhen",
    success : function(data, textStatus) {
                    alert(data);
  alert(this.success);
      }
});
```

$.get(url, data, callback, type)，$.ajax()的简易方法，用于发送 GET 请求

url：请求地址
data：发送到服务端的请求参数
callback：请求成功后的回调函数，函数原型是：
function (data, textStatus) {
    // data could be xmlDoc, jsonObj, html, text, etc...
    this; // the options for this ajax request
}

$.post(url, data, callback, type)，$.ajax()的简易方法，跟$.get()差不多，用于发送 POST 请求

23、浏览器及特性检测

$.support，可以检测当前浏览器是否支持下列属性，返回 boolean。包括 boxModel、cssFloat、opacity、tbody 等
$.browser，检测当前浏览器类型，返回一个 map，其中可能的值有 safari、opera、msie、mozilla

24、数据缓存方法

该类方法是 jQuery.data()方法和 jQuery.removeData()的另一种形式，增加的 elem 参数是 DOM Element

$.data(elem, name)，取出 elem 上 name 的值
$.data(elem, name, value)，设置 elem 上 name 的值为 value
$.removeData(elem, name)，删除 elem 上的 name
$.removeData(elem)，删除 elem 上的所有缓存数据

25、工具方法

$.isArray(obj)，检测一个对象是否是数组
$.isFunction(obj)，检测一个对象是否是函数
$.trim(str)，去除 string 的空格
$.inArray(value, array)，返回 value 在 array 中的下标，如果没有找到则返回-1，比如 $.inArray(123, ["john",1,123,"f"])将会返回 2
$.unique(array)，去除 array 中的重复元素，该方法只对 DOM Element 有效，对 string 和 number 无效