

Evan

Only let oneself become strong enough, good enough, can afford the life that you want to.

目录视图

摘要视图

RSS 订阅

管理博文

文章

评论送书 | 云原生、Docker、Web算法 为什么我们创业失败了和选择创业公司的思考 福利 | 免费参加 2017 OpenStack Days China

Jquery中find与each方法使用详解

标签：jquery find each

2017-02-11 13:24 138人阅读 评论(0) 收藏 编辑 删除

分类： JQuery (9)

目录(?) [+]

本文实例讲述了jQuery中find与each方法用法。分享给大家供大家参考。具体如下：

一、find()方法

jQuery选择器非常强大，利用css的命名规约，可以更快更方便的找出想要的元素。

图解：

```
<section>
  <input type="hidden" name="productPicture" value="{property.productPicture}">
  <input type="hidden" name="productPrice" value="{property.productPrice}">
  <input type="hidden" name="productStock" value="{property.productStock}">
  <input type="hidden" name="propertyName" value="{property.propertyName}">
  {property.propertyName}
</section>

$('.type-button-detail').find('section').click(function(){
  var productPicture = $(this).find("input[name='productPicture']").val();
  var productPrice = $(this).find("input[name='productPrice']").val();
  var productStock = $(this).find("input[name='productStock']").val();
  var propertyName = $(this).find("input[name='propertyName']").val();
  alert(productPicture+" "+productPrice+" "+productStock+" "+propertyName);

  var price=Number($('.ui-currency-sign.type-00').text());
  var index=$(this).index();
  var $type=$(this).text();p://blog.csdn.net/molashaonian
  var src=$(.main-image').attr('src'); //读取商品主图

  $(this).toggleClass('choosed').siblings().removeClass('choosed');
  if($(this).hasClass('choosed')){
    $('.display-choosed,choose-product-type').html('已选择 '+propertyName);
    $('.storage-count').find('span').html(productStock);
    $('.display-price').find('span').html(productPrice);
    $('.type-display-image').find('img').attr('src',productPicture);
  }else{
    $('.display-choosed,choose-product-type').html('选择 属性分类');
    $('.storage-count').find('span').html(100);
    $('.display-price').find('span').html(price);
    $('.type-display-image').find('img').attr('src',src);
  }
})
```

比如：

```
[javascript]
01. <span style="font-size:18px;">
02.
03. $("#id")
```

```
04.    $("#"+id")
05.    $(this)
06.    $(element)
07.
08.
09.    </span>
```

等等，只要灵活运用，就能爆发出强大的可造型。

但是在实际使用中，仍然觉得有些不足。

如果想要在某个元素下寻找特定的元素，仅仅依靠上面这个方法，就必须对 \$("#id")获取的元素进行遍历，获取其子元素。如此一来就显得格外的繁琐，代码量也非常庞大。

于是这就需要用到find()方法。

[javascript]

```
01.    <span style="font-size:18px;">
02.
03.    $("#id").find("#child");
04.    $("#id").find(".child");
05.    $("#id").find("input[type='image']");
06.
07.
08.    </span>
```

非常方便好用。

除了上面的find()方法之外，还有一种查找子元素的方法。

[javascript]

```
01.    <span style="font-size:18px;">
02.
03.    $(".child",parent);
04.
05.
06.    </span>
```

这种方法与find()方法的结果是一样的，也更加简洁。

我们举个例子：

[javascript]

```
01.    <span style="font-size:18px;">
02.
03.    function(table){
04.        $("tr",table).css("background-color","red");
05.    }
06.
07.
08.    </span>
```

这种方法，方便代码的重用，更符合闭包的写法。

二、each()方法

each()方法能使DOM循环结构简洁，不容易出错。each()函数封装了十分强大的遍历功能，使用也很方便，可以遍历一维数组、多维数组、DOM, JSON 等等

在JavaScript开发过程中使用\$each可以大大的减轻我们的工作量。

下面提一下each的几种常用的用法

each处理一维数组

```
[javascript]
01. <span style="font-size:18px;">
02.
03. var arr1 = [ "aaa", "bbb", "ccc" ];
04. $.each(arr1, function(i,val){
05.     alert(i);
06.     alert(val);
07. });
08.
09.
10. </span>
```

alert(i)将输出0, 1, 2

alert(val)将输出aaa, bbb, ccc

each处理二维数组

```
[javascript]
01. <span style="font-size:18px;">
02.     var arr2 = [['a', 'aa', 'aaa'], ['b', 'bb', 'bbb'], ['c', 'cc', 'ccc']]
03.     $.each(arr, function(i, item){
04.         alert(i);
05.         alert(item);
06.     });
07.
08.
09. </span>
```

arr2为一个二维数组，item相当于取这二维数组中的每一个数组。

item[0]相对于取每一个一维数组里的第一个值

alert(i)将输出为0, 1, 2, 因为这二维数组含有3个数组元素

alert(item)将输出为 ['a', 'aa', 'aaa'], ['b', 'bb', 'bbb'], ['c', 'cc', 'ccc']

对此二位数组的处理稍作变更之后

```
[javascript]
01. <span style="font-size:18px;">
02.
03.     var arr = [['a', 'aa', 'aaa'], ['b', 'bb', 'bbb'], ['c', 'cc', 'ccc']]
04.     $.each(arr, function(i, item){
05.         $.each(item,function(j,val){
06.             alert(j);
07.             alert(val);
```

```
08.         });  
09.     });  
10.  
11.  
12.  
13. </span>
```

alert(j)将输出为0, 1, 2, 0, 1, 2, 0, 1, 2

alert(val)将输出为a, aa, aaa, b, bb, bbb, c, cc, ccc

each处理json数据, 这个each就有更厉害了, 能循环每一个属性

```
[javascript]  
01. <span style="font-size:18px;">  
02.  
03.  
04. var obj = { one:1, two:2, three:3};  
05.     each(obj, function(key, val) {  
06.         alert(key);  
07.         alert(val);  
08.     });  
09.  
10.  
11.  
12. </span>
```

这里alert(key)将输出one two three

alert(val)将输出one, 1, two, 2, three,3

这边为何key不是数字而是属性呢, 因为json格式内是一组无序的属性-值, 既然无序, 又何来数字呢。而这个val等同于obj[key]

each处理dom元素, 此处以一个input表单元素作为例子。

如果你dom中有一段这样的代码

```
[javascript]  
01. <span style="font-size:18px;">  
02.  
03.     <input name="aaa" type="hidden" value="111" />  
04.     <input name="bbb" type="hidden" value="222" />  
05.     <input name="ccc" type="hidden" value="333" />  
06.     <input name="ddd" type="hidden" value="444"/>  
07.  
08.  
09. </span>
```

然后你使用each如下

```
[javascript]  
01. <span style="font-size:18px;">  
02.  
03. $.each($(".input:hidden"), function(i,val){  
04.     alert(val);  
05.     alert(i);  
06.     alert(val.name);  
07. });  
08. </span>
```

```
07.         alert(val.value);
08.     });
09.
10.
11. </span>
```

那么，`alert(val)`将输出[object HTMLInputElement]，因为它是一个表单元素。

`alert(i)`将输出为0，1，2，3

`alert(val.name)`；将输出aaa,bbb,ccc,ddd，如果使用`this.name`将输出同样的结果

`alert(val.value)`；将输出111,222,333,444，如果使用`this.value`将输出同样的结果

如果将以上面一段代码改变成如下的形式

```
[javascript]
01. <span style="font-size:18px;">
02.
03.     $("input:hidden").each(function(i,val){
04.         alert(i);
05.         alert(val.name);
06.         alert(val.value);
07.     });
08.
09.
10. </span>
```

可以看到，输出的结果是一样的，至于两种写法究竟区别在哪，我也还不知。此改变运用到上面几段数组的操作也会输出同样的结果。

这样，几个例子的实际结果已经得到答案。接着再继续往下研究，总不能知其然不知其所以然。

从以上的例子中可知jQuery和jQuery对象都实现了该方法，对于jQuery对象，只是把each方法简单的进行了委托：把jQuery对象作为第一个参数传递给jQuery的each方法。

看下jQuery中的each实现（网络摘抄）

```
[javascript]
01. <span style="font-size:18px;">
02.
03.     function (object, callback, args) {
04.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
05.         var name, i = 0, length = object.length;
06.         if (args) {
07.             if (length == undefined) {
08.                 for (name in object) {
09.                     if (callback.apply(object[name], args) === false) {
10.                         break;
11.                     }
12.                 }
13.             } else {
14.                 for (; i < length; i++) {
15.                     if (callback.apply(object[i++], args) === false) {
16.                         break;
17.                     }
18.                 }
19.             }
20.         } else {
21.             if (length == undefined) {
22.                 for (name in object) {
23.                     if (callback.call(object[name], name, object[name]) === false) {
24.                         break;
25.                     }
26.                 }
27.             } else {
28.                 for (; i < length; i++) {
29.                     if (callback.call(object[i], i, object[i]) === false) {
30.                         break;
31.                     }
32.                 }
33.             }
34.         }
35.     }
36.
37.     function (object, callback, args) {
38.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
39.         var name, i = 0, length = object.length;
40.         if (args) {
41.             if (length == undefined) {
42.                 for (name in object) {
43.                     if (callback.apply(object[name], args) === false) {
44.                         break;
45.                     }
46.                 }
47.             } else {
48.                 for (; i < length; i++) {
49.                     if (callback.apply(object[i++], args) === false) {
50.                         break;
51.                     }
52.                 }
53.             }
54.         } else {
55.             if (length == undefined) {
56.                 for (name in object) {
57.                     if (callback.call(object[name], name, object[name]) === false) {
58.                         break;
59.                     }
60.                 }
61.             } else {
62.                 for (; i < length; i++) {
63.                     if (callback.call(object[i], i, object[i]) === false) {
64.                         break;
65.                     }
66.                 }
67.             }
68.         }
69.     }
70.
71.     function (object, callback, args) {
72.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
73.         var name, i = 0, length = object.length;
74.         if (args) {
75.             if (length == undefined) {
76.                 for (name in object) {
77.                     if (callback.apply(object[name], args) === false) {
78.                         break;
79.                     }
80.                 }
81.             } else {
82.                 for (; i < length; i++) {
83.                     if (callback.apply(object[i++], args) === false) {
84.                         break;
85.                     }
86.                 }
87.             }
88.         } else {
89.             if (length == undefined) {
90.                 for (name in object) {
91.                     if (callback.call(object[name], name, object[name]) === false) {
92.                         break;
93.                     }
94.                 }
95.             } else {
96.                 for (; i < length; i++) {
97.                     if (callback.call(object[i], i, object[i]) === false) {
98.                         break;
99.                     }
100.                 }
101.             }
102.         }
103.     }
104.
105.     function (object, callback, args) {
106.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
107.         var name, i = 0, length = object.length;
108.         if (args) {
109.             if (length == undefined) {
110.                 for (name in object) {
111.                     if (callback.apply(object[name], args) === false) {
112.                         break;
113.                     }
114.                 }
115.             } else {
116.                 for (; i < length; i++) {
117.                     if (callback.apply(object[i++], args) === false) {
118.                         break;
119.                     }
120.                 }
121.             }
122.         } else {
123.             if (length == undefined) {
124.                 for (name in object) {
125.                     if (callback.call(object[name], name, object[name]) === false) {
126.                         break;
127.                     }
128.                 }
129.             } else {
130.                 for (; i < length; i++) {
131.                     if (callback.call(object[i], i, object[i]) === false) {
132.                         break;
133.                     }
134.                 }
135.             }
136.         }
137.     }
138.
139.     function (object, callback, args) {
140.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
141.         var name, i = 0, length = object.length;
142.         if (args) {
143.             if (length == undefined) {
144.                 for (name in object) {
145.                     if (callback.apply(object[name], args) === false) {
146.                         break;
147.                     }
148.                 }
149.             } else {
150.                 for (; i < length; i++) {
151.                     if (callback.apply(object[i++], args) === false) {
152.                         break;
153.                     }
154.                 }
155.             }
156.         } else {
157.             if (length == undefined) {
158.                 for (name in object) {
159.                     if (callback.call(object[name], name, object[name]) === false) {
160.                         break;
161.                     }
162.                 }
163.             } else {
164.                 for (; i < length; i++) {
165.                     if (callback.call(object[i], i, object[i]) === false) {
166.                         break;
167.                     }
168.                 }
169.             }
170.         }
171.     }
172.
173.     function (object, callback, args) {
174.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
175.         var name, i = 0, length = object.length;
176.         if (args) {
177.             if (length == undefined) {
178.                 for (name in object) {
179.                     if (callback.apply(object[name], args) === false) {
180.                         break;
181.                     }
182.                 }
183.             } else {
184.                 for (; i < length; i++) {
185.                     if (callback.apply(object[i++], args) === false) {
186.                         break;
187.                     }
188.                 }
189.             }
190.         } else {
191.             if (length == undefined) {
192.                 for (name in object) {
193.                     if (callback.call(object[name], name, object[name]) === false) {
194.                         break;
195.                     }
196.                 }
197.             } else {
198.                 for (; i < length; i++) {
199.                     if (callback.call(object[i], i, object[i]) === false) {
200.                         break;
201.                     }
202.                 }
203.             }
204.         }
205.     }
206.
207.     function (object, callback, args) {
208.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
209.         var name, i = 0, length = object.length;
210.         if (args) {
211.             if (length == undefined) {
212.                 for (name in object) {
213.                     if (callback.apply(object[name], args) === false) {
214.                         break;
215.                     }
216.                 }
217.             } else {
218.                 for (; i < length; i++) {
219.                     if (callback.apply(object[i++], args) === false) {
220.                         break;
221.                     }
222.                 }
223.             }
224.         } else {
225.             if (length == undefined) {
226.                 for (name in object) {
227.                     if (callback.call(object[name], name, object[name]) === false) {
228.                         break;
229.                     }
230.                 }
231.             } else {
232.                 for (; i < length; i++) {
233.                     if (callback.call(object[i], i, object[i]) === false) {
234.                         break;
235.                     }
236.                 }
237.             }
238.         }
239.     }
240.
241.     function (object, callback, args) {
242.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
243.         var name, i = 0, length = object.length;
244.         if (args) {
245.             if (length == undefined) {
246.                 for (name in object) {
247.                     if (callback.apply(object[name], args) === false) {
248.                         break;
249.                     }
250.                 }
251.             } else {
252.                 for (; i < length; i++) {
253.                     if (callback.apply(object[i++], args) === false) {
254.                         break;
255.                     }
256.                 }
257.             }
258.         } else {
259.             if (length == undefined) {
260.                 for (name in object) {
261.                     if (callback.call(object[name], name, object[name]) === false) {
262.                         break;
263.                     }
264.                 }
265.             } else {
266.                 for (; i < length; i++) {
267.                     if (callback.call(object[i], i, object[i]) === false) {
268.                         break;
269.                     }
270.                 }
271.             }
272.         }
273.     }
274.
275.     function (object, callback, args) {
276.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
277.         var name, i = 0, length = object.length;
278.         if (args) {
279.             if (length == undefined) {
280.                 for (name in object) {
281.                     if (callback.apply(object[name], args) === false) {
282.                         break;
283.                     }
284.                 }
285.             } else {
286.                 for (; i < length; i++) {
287.                     if (callback.apply(object[i++], args) === false) {
288.                         break;
289.                     }
290.                 }
291.             }
292.         } else {
293.             if (length == undefined) {
294.                 for (name in object) {
295.                     if (callback.call(object[name], name, object[name]) === false) {
296.                         break;
297.                     }
298.                 }
299.             } else {
300.                 for (; i < length; i++) {
301.                     if (callback.call(object[i], i, object[i]) === false) {
302.                         break;
303.                     }
304.                 }
305.             }
306.         }
307.     }
308.
309.     function (object, callback, args) {
310.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
311.         var name, i = 0, length = object.length;
312.         if (args) {
313.             if (length == undefined) {
314.                 for (name in object) {
315.                     if (callback.apply(object[name], args) === false) {
316.                         break;
317.                     }
318.                 }
319.             } else {
320.                 for (; i < length; i++) {
321.                     if (callback.apply(object[i++], args) === false) {
322.                         break;
323.                     }
324.                 }
325.             }
326.         } else {
327.             if (length == undefined) {
328.                 for (name in object) {
329.                     if (callback.call(object[name], name, object[name]) === false) {
330.                         break;
331.                     }
332.                 }
333.             } else {
334.                 for (; i < length; i++) {
335.                     if (callback.call(object[i], i, object[i]) === false) {
336.                         break;
337.                     }
338.                 }
339.             }
340.         }
341.     }
342.
343.     function (object, callback, args) {
344.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
345.         var name, i = 0, length = object.length;
346.         if (args) {
347.             if (length == undefined) {
348.                 for (name in object) {
349.                     if (callback.apply(object[name], args) === false) {
350.                         break;
351.                     }
352.                 }
353.             } else {
354.                 for (; i < length; i++) {
355.                     if (callback.apply(object[i++], args) === false) {
356.                         break;
357.                     }
358.                 }
359.             }
360.         } else {
361.             if (length == undefined) {
362.                 for (name in object) {
363.                     if (callback.call(object[name], name, object[name]) === false) {
364.                         break;
365.                     }
366.                 }
367.             } else {
368.                 for (; i < length; i++) {
369.                     if (callback.call(object[i], i, object[i]) === false) {
370.                         break;
371.                     }
372.                 }
373.             }
374.         }
375.     }
376.
377.     function (object, callback, args) {
378.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
379.         var name, i = 0, length = object.length;
380.         if (args) {
381.             if (length == undefined) {
382.                 for (name in object) {
383.                     if (callback.apply(object[name], args) === false) {
384.                         break;
385.                     }
386.                 }
387.             } else {
388.                 for (; i < length; i++) {
389.                     if (callback.apply(object[i++], args) === false) {
390.                         break;
391.                     }
392.                 }
393.             }
394.         } else {
395.             if (length == undefined) {
396.                 for (name in object) {
397.                     if (callback.call(object[name], name, object[name]) === false) {
398.                         break;
399.                     }
400.                 }
401.             } else {
402.                 for (; i < length; i++) {
403.                     if (callback.call(object[i], i, object[i]) === false) {
404.                         break;
405.                     }
406.                 }
407.             }
408.         }
409.     }
410.
411.     function (object, callback, args) {
412.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
413.         var name, i = 0, length = object.length;
414.         if (args) {
415.             if (length == undefined) {
416.                 for (name in object) {
417.                     if (callback.apply(object[name], args) === false) {
418.                         break;
419.                     }
420.                 }
421.             } else {
422.                 for (; i < length; i++) {
423.                     if (callback.apply(object[i++], args) === false) {
424.                         break;
425.                     }
426.                 }
427.             }
428.         } else {
429.             if (length == undefined) {
430.                 for (name in object) {
431.                     if (callback.call(object[name], name, object[name]) === false) {
432.                         break;
433.                     }
434.                 }
435.             } else {
436.                 for (; i < length; i++) {
437.                     if (callback.call(object[i], i, object[i]) === false) {
438.                         break;
439.                     }
440.                 }
441.             }
442.         }
443.     }
444.
445.     function (object, callback, args) {
446.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
447.         var name, i = 0, length = object.length;
448.         if (args) {
449.             if (length == undefined) {
450.                 for (name in object) {
451.                     if (callback.apply(object[name], args) === false) {
452.                         break;
453.                     }
454.                 }
455.             } else {
456.                 for (; i < length; i++) {
457.                     if (callback.apply(object[i++], args) === false) {
458.                         break;
459.                     }
460.                 }
461.             }
462.         } else {
463.             if (length == undefined) {
464.                 for (name in object) {
465.                     if (callback.call(object[name], name, object[name]) === false) {
466.                         break;
467.                     }
468.                 }
469.             } else {
470.                 for (; i < length; i++) {
471.                     if (callback.call(object[i], i, object[i]) === false) {
472.                         break;
473.                     }
474.                 }
475.             }
476.         }
477.     }
478.
479.     function (object, callback, args) {
480.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
481.         var name, i = 0, length = object.length;
482.         if (args) {
483.             if (length == undefined) {
484.                 for (name in object) {
485.                     if (callback.apply(object[name], args) === false) {
486.                         break;
487.                     }
488.                 }
489.             } else {
490.                 for (; i < length; i++) {
491.                     if (callback.apply(object[i++], args) === false) {
492.                         break;
493.                     }
494.                 }
495.             }
496.         } else {
497.             if (length == undefined) {
498.                 for (name in object) {
499.                     if (callback.call(object[name], name, object[name]) === false) {
500.                         break;
501.                     }
502.                 }
503.             } else {
504.                 for (; i < length; i++) {
505.                     if (callback.call(object[i], i, object[i]) === false) {
506.                         break;
507.                     }
508.                 }
509.             }
510.         }
511.     }
512.
513.     function (object, callback, args) {
514.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
515.         var name, i = 0, length = object.length;
516.         if (args) {
517.             if (length == undefined) {
518.                 for (name in object) {
519.                     if (callback.apply(object[name], args) === false) {
520.                         break;
521.                     }
522.                 }
523.             } else {
524.                 for (; i < length; i++) {
525.                     if (callback.apply(object[i++], args) === false) {
526.                         break;
527.                     }
528.                 }
529.             }
530.         } else {
531.             if (length == undefined) {
532.                 for (name in object) {
533.                     if (callback.call(object[name], name, object[name]) === false) {
534.                         break;
535.                     }
536.                 }
537.             } else {
538.                 for (; i < length; i++) {
539.                     if (callback.call(object[i], i, object[i]) === false) {
540.                         break;
541.                     }
542.                 }
543.             }
544.         }
545.     }
546.
547.     function (object, callback, args) {
548.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
549.         var name, i = 0, length = object.length;
550.         if (args) {
551.             if (length == undefined) {
552.                 for (name in object) {
553.                     if (callback.apply(object[name], args) === false) {
554.                         break;
555.                     }
556.                 }
557.             } else {
558.                 for (; i < length; i++) {
559.                     if (callback.apply(object[i++], args) === false) {
560.                         break;
561.                     }
562.                 }
563.             }
564.         } else {
565.             if (length == undefined) {
566.                 for (name in object) {
567.                     if (callback.call(object[name], name, object[name]) === false) {
568.                         break;
569.                     }
570.                 }
571.             } else {
572.                 for (; i < length; i++) {
573.                     if (callback.call(object[i], i, object[i]) === false) {
574.                         break;
575.                     }
576.                 }
577.             }
578.         }
579.     }
580.
581.     function (object, callback, args) {
582.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
583.         var name, i = 0, length = object.length;
584.         if (args) {
585.             if (length == undefined) {
586.                 for (name in object) {
587.                     if (callback.apply(object[name], args) === false) {
588.                         break;
589.                     }
590.                 }
591.             } else {
592.                 for (; i < length; i++) {
593.                     if (callback.apply(object[i++], args) === false) {
594.                         break;
595.                     }
596.                 }
597.             }
598.         } else {
599.             if (length == undefined) {
600.                 for (name in object) {
601.                     if (callback.call(object[name], name, object[name]) === false) {
602.                         break;
603.                     }
604.                 }
605.             } else {
606.                 for (; i < length; i++) {
607.                     if (callback.call(object[i], i, object[i]) === false) {
608.                         break;
609.                     }
610.                 }
611.             }
612.         }
613.     }
614.
615.     function (object, callback, args) {
616.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
617.         var name, i = 0, length = object.length;
618.         if (args) {
619.             if (length == undefined) {
620.                 for (name in object) {
621.                     if (callback.apply(object[name], args) === false) {
622.                         break;
623.                     }
624.                 }
625.             } else {
626.                 for (; i < length; i++) {
627.                     if (callback.apply(object[i++], args) === false) {
628.                         break;
629.                     }
630.                 }
631.             }
632.         } else {
633.             if (length == undefined) {
634.                 for (name in object) {
635.                     if (callback.call(object[name], name, object[name]) === false) {
636.                         break;
637.                     }
638.                 }
639.             } else {
640.                 for (; i < length; i++) {
641.                     if (callback.call(object[i], i, object[i]) === false) {
642.                         break;
643.                     }
644.                 }
645.             }
646.         }
647.     }
648.
649.     function (object, callback, args) {
650.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
651.         var name, i = 0, length = object.length;
652.         if (args) {
653.             if (length == undefined) {
654.                 for (name in object) {
655.                     if (callback.apply(object[name], args) === false) {
656.                         break;
657.                     }
658.                 }
659.             } else {
660.                 for (; i < length; i++) {
661.                     if (callback.apply(object[i++], args) === false) {
662.                         break;
663.                     }
664.                 }
665.             }
666.         } else {
667.             if (length == undefined) {
668.                 for (name in object) {
669.                     if (callback.call(object[name], name, object[name]) === false) {
670.                         break;
671.                     }
672.                 }
673.             } else {
674.                 for (; i < length; i++) {
675.                     if (callback.call(object[i], i, object[i]) === false) {
676.                         break;
677.                     }
678.                 }
679.             }
680.         }
681.     }
682.
683.     function (object, callback, args) {
684.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
685.         var name, i = 0, length = object.length;
686.         if (args) {
687.             if (length == undefined) {
688.                 for (name in object) {
689.                     if (callback.apply(object[name], args) === false) {
690.                         break;
691.                     }
692.                 }
693.             } else {
694.                 for (; i < length; i++) {
695.                     if (callback.apply(object[i++], args) === false) {
696.                         break;
697.                     }
698.                 }
699.             }
700.         } else {
701.             if (length == undefined) {
702.                 for (name in object) {
703.                     if (callback.call(object[name], name, object[name]) === false) {
704.                         break;
705.                     }
706.                 }
707.             } else {
708.                 for (; i < length; i++) {
709.                     if (callback.call(object[i], i, object[i]) === false) {
710.                         break;
711.                     }
712.                 }
713.             }
714.         }
715.     }
716.
717.     function (object, callback, args) {
718.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
719.         var name, i = 0, length = object.length;
720.         if (args) {
721.             if (length == undefined) {
722.                 for (name in object) {
723.                     if (callback.apply(object[name], args) === false) {
724.                         break;
725.                     }
726.                 }
727.             } else {
728.                 for (; i < length; i++) {
729.                     if (callback.apply(object[i++], args) === false) {
730.                         break;
731.                     }
732.                 }
733.             }
734.         } else {
735.             if (length == undefined) {
736.                 for (name in object) {
737.                     if (callback.call(object[name], name, object[name]) === false) {
738.                         break;
739.                     }
740.                 }
741.             } else {
742.                 for (; i < length; i++) {
743.                     if (callback.call(object[i], i, object[i]) === false) {
744.                         break;
745.                     }
746.                 }
747.             }
748.         }
749.     }
750.
751.     function (object, callback, args) {
752.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
753.         var name, i = 0, length = object.length;
754.         if (args) {
755.             if (length == undefined) {
756.                 for (name in object) {
757.                     if (callback.apply(object[name], args) === false) {
758.                         break;
759.                     }
760.                 }
761.             } else {
762.                 for (; i < length; i++) {
763.                     if (callback.apply(object[i++], args) === false) {
764.                         break;
765.                     }
766.                 }
767.             }
768.         } else {
769.             if (length == undefined) {
770.                 for (name in object) {
771.                     if (callback.call(object[name], name, object[name]) === false) {
772.                         break;
773.                     }
774.                 }
775.             } else {
776.                 for (; i < length; i++) {
777.                     if (callback.call(object[i], i, object[i]) === false) {
778.                         break;
779.                     }
780.                 }
781.             }
782.         }
783.     }
784.
785.     function (object, callback, args) {
786.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
787.         var name, i = 0, length = object.length;
788.         if (args) {
789.             if (length == undefined) {
790.                 for (name in object) {
791.                     if (callback.apply(object[name], args) === false) {
792.                         break;
793.                     }
794.                 }
795.             } else {
796.                 for (; i < length; i++) {
797.                     if (callback.apply(object[i++], args) === false) {
798.                         break;
799.                     }
800.                 }
801.             }
802.         } else {
803.             if (length == undefined) {
804.                 for (name in object) {
805.                     if (callback.call(object[name], name, object[name]) === false) {
806.                         break;
807.                     }
808.                 }
809.             } else {
810.                 for (; i < length; i++) {
811.                     if (callback.call(object[i], i, object[i]) === false) {
812.                         break;
813.                     }
814.                 }
815.             }
816.         }
817.     }
818.
819.     function (object, callback, args) {
820.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
821.         var name, i = 0, length = object.length;
822.         if (args) {
823.             if (length == undefined) {
824.                 for (name in object) {
825.                     if (callback.apply(object[name], args) === false) {
826.                         break;
827.                     }
828.                 }
829.             } else {
830.                 for (; i < length; i++) {
831.                     if (callback.apply(object[i++], args) === false) {
832.                         break;
833.                     }
834.                 }
835.             }
836.         } else {
837.             if (length == undefined) {
838.                 for (name in object) {
839.                     if (callback.call(object[name], name, object[name]) === false) {
840.                         break;
841.                     }
842.                 }
843.             } else {
844.                 for (; i < length; i++) {
845.                     if (callback.call(object[i], i, object[i]) === false) {
846.                         break;
847.                     }
848.                 }
849.             }
850.         }
851.     }
852.
853.     function (object, callback, args) {
854.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
855.         var name, i = 0, length = object.length;
856.         if (args) {
857.             if (length == undefined) {
858.                 for (name in object) {
859.                     if (callback.apply(object[name], args) === false) {
860.                         break;
861.                     }
862.                 }
863.             } else {
864.                 for (; i < length; i++) {
865.                     if (callback.apply(object[i++], args) === false) {
866.                         break;
867.                     }
868.                 }
869.             }
870.         } else {
871.             if (length == undefined) {
872.                 for (name in object) {
873.                     if (callback.call(object[name], name, object[name]) === false) {
874.                         break;
875.                     }
876.                 }
877.             } else {
878.                 for (; i < length; i++) {
879.                     if (callback.call(object[i], i, object[i]) === false) {
880.                         break;
881.                     }
882.                 }
883.             }
884.         }
885.     }
886.
887.     function (object, callback, args) {
888.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
889.         var name, i = 0, length = object.length;
890.         if (args) {
891.             if (length == undefined) {
892.                 for (name in object) {
893.                     if (callback.apply(object[name], args) === false) {
894.                         break;
895.                     }
896.                 }
897.             } else {
898.                 for (; i < length; i++) {
899.                     if (callback.apply(object[i++], args) === false) {
900.                         break;
901.                     }
902.                 }
903.             }
904.         } else {
905.             if (length == undefined) {
906.                 for (name in object) {
907.                     if (callback.call(object[name], name, object[name]) === false) {
908.                         break;
909.                     }
910.                 }
911.             } else {
912.                 for (; i < length; i++) {
913.                     if (callback.call(object[i], i, object[i]) === false) {
914.                         break;
915.                     }
916.                 }
917.             }
918.         }
919.     }
920.
921.     function (object, callback, args) {
922.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
923.         var name, i = 0, length = object.length;
924.         if (args) {
925.             if (length == undefined) {
926.                 for (name in object) {
927.                     if (callback.apply(object[name], args) === false) {
928.                         break;
929.                     }
930.                 }
931.             } else {
932.                 for (; i < length; i++) {
933.                     if (callback.apply(object[i++], args) === false) {
934.                         break;
935.                     }
936.                 }
937.             }
938.         } else {
939.             if (length == undefined) {
940.                 for (name in object) {
941.                     if (callback.call(object[name], name, object[name]) === false) {
942.                         break;
943.                     }
944.                 }
945.             } else {
946.                 for (; i < length; i++) {
947.                     if (callback.call(object[i], i, object[i]) === false) {
948.                         break;
949.                     }
950.                 }
951.             }
952.         }
953.     }
954.
955.     function (object, callback, args) {
956.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
957.         var name, i = 0, length = object.length;
958.         if (args) {
959.             if (length == undefined) {
960.                 for (name in object) {
961.                     if (callback.apply(object[name], args) === false) {
962.                         break;
963.                     }
964.                 }
965.             } else {
966.                 for (; i < length; i++) {
967.                     if (callback.apply(object[i++], args) === false) {
968.                         break;
969.                     }
970.                 }
971.             }
972.         } else {
973.             if (length == undefined) {
974.                 for (name in object) {
975.                     if (callback.call(object[name], name, object[name]) === false) {
976.                         break;
977.                     }
978.                 }
979.             } else {
980.                 for (; i < length; i++) {
981.                     if (callback.call(object[i], i, object[i]) === false) {
982.                         break;
983.                     }
984.                 }
985.             }
986.         }
987.     }
988.
989.     function (object, callback, args) {
990.         //该方法有三个参数:进行操作的对象obj, 进行操作的函数fn, 函数的参数args
991.         var name, i = 0, length = object.length;
992.         if (args) {
993.             if (length == undefined) {
994.                 for (name in object) {
995.                     if (callback.apply(object[name], args) === false) {
996.                         break;
997.                     }
998.                 }
999.             } else {
1000.                 for (; i < length; i++) {
1001.                     if (callback.apply(object[i++], args) === false) {
1002.                         break;
1003.                     }
1004.                 }
1005.             }
1006.         } else {
1007.             if (length == undefined) {
1008.                 for (name in object) {
1009.                     if (callback.call(object[name], name, object[name]) === false) {
1010.                         break;
1011.                     }
1012.                 }
1013.             } else {
1014.                 for (; i < length; i++) {
1015.                     if (callback.call(object[i], i, object[i]) === false) {
1016.                         break;
1017.                     }
1018.                 }
1019.             }
1020.         }
1021.     }
1022.
1023.     function (object, callback, args) {
1024.         //该方法有三个
```

```
25.     }
26.     }
27.   } else {
28.     for (var value = object[0]; i < length && callback.call(value, i, value) !== false; value = object[++i]) {}
29.     /*object[0]取得jQuery对象中的第一个DOM元素，通过for循环，
30.     得到遍历整个jQuery对象中对应的每个DOM元素，通过 callback.call( value,i,value);
31.     将callback的this对象指向value对象，并且传递两个参数,i表示索引值，value表示DOM元素；
32.     其中callback是类似于 function(index, elem) { ... } 的方法。
33.     所以就得到 $("....").each(function(index, elem){ ... });
34.     */
35.   }
36. }
37. return object;
38. }
39.
40.
41.
42. </span>
```

jquery会自动根据传入的元素进行判断，然后在根据判断结果采取apply还是call方法的处理。在fn的实现中，可以直接采用this指针引用数组或是对象的子元素。

1.obj对象是数组

each方法会对数组中子元素的逐个进行fn函数调用，直至调用某个子元素返回的结果为false为止，也就是说，我们可以在提供的fn函数进行处理，使之满足一定条件后就退出each方法调用。当each方法提供了arg参数时，fn函数调用传入的参数为arg，否则为：子元素索引，子元素本身

2.obj 对象不是数组

该方法同1的最大区别是：fn方法会被逐次不考虑返回值的进行进行。换句话说，obj对象的所有属性都会被fn方法进行调用，即使fn函数返回false。调用传入的参数同1类似。

≡ eq() 定义和用法

eq() 方法将匹配元素集缩减值指定 index 上的一个。

语法

`.eq(index)`

参数	描述
index	整数，指示元素的位置（最小为 0）。 如果是负数，则从集合中的最后一个元素往回计数。

详细说明

如果给定表示 DOM 元素集合的 **iaquerv** 对象，.eq() 方法会用集合中的一个元素构造一个新的 jQuery 对象。所使用的 index 参数标示集合中元素的位置。

请看下面这个简单的列表：

```
<ul>
  <li>list item 1</li>
  <li>list item 2</li>
  <li>list item 3</li>
  <li>list item 4</li>
```

```
<li>list item 5</li>
</ul>
```

例子 1

我们可以把该方法应用到这个列表项目集:

```
$('li').eq(2).css('background-color', 'red');
```

这个调用的结果是项目 3 设置了红色背景。请注意, `index` 是基于零的, 并且是在 `jQuery` 对象中引用元素的位置, 而不是在 `DOM` 树中。

综合的例子

[javascript]

```
01. <tbody id="ScoreTab">
02.   <tr>
03.     <td>陈晶/chenjing1</td>
04.     <td>
05.       <input type="radio" name="grade0" value="5">5分
06.       <input type="radio" name="grade0" value="4">4分
07.       <input type="radio" name="grade0" value="3">3分
08.       <input type="radio" name="grade0" value="2">2分
09.       <input type="radio" name="grade0" value="1">1分
10.       <span style="display:none">e74dde12-68a3-36d6-8cac-a778186053fd</span>
11.     </td>
12.   </tr>
13.   <tr>
14.     <td>张硕/zhangshuo</td>
15.     <td>
16.       <input type="radio" name="grade1" value="5">5分
17.       <input type="radio" name="grade1" value="4">4分
18.       <input type="radio" name="grade1" value="3">3分
19.       <input type="radio" name="grade1" value="2">2分
20.       <input type="radio" name="grade1" value="1">1分
21.       <span style="display:none">a456efa0-af65-31ff-9646-26a8800e77cb</span>
22.     </td>
23.   </tr>
24. </tbody>
```

代码

[javascript]

```
01. <span style="font-size:18px;">
02.
03.   function getScore(opt)
04.   {
05.     var x=0;
06.     //获取 <tbody id="ScoreTab">下面的所有tr
07.     $("#ScoreTab").find("tr").each(function(){
08.       //获取tr下面的所有子元素
09.       var arrtd = $(this).children();
10.       var psn = arrtd.eq(0).text();
11.       //获取第二个子元素下面 input元素
12.       var scoreInput = arrtd.eq(1).find("input");
13.       //获取单选框选中的值
14.       var scoreNum = $("input[name='grade'+x+'']:checked").val();
15.       var psn = arrtd.eq(1).find("span").text();
16.       x++;
17.     })
18.   }
19.
20.
21.
```