jQuery快速入门

概述

> jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.

jQuery是一个快速、简洁、具有丰富特性的JavaScript库。通过简单易用的API，兼容各主流浏览器，并使得HTML文档操作、事件处理、动画、Ajax交互等事情变得更加简单。

- 快速获取文档元素
  - jQuery提供了快速查询DOM文档中元素的能力，而且强化了JavaScript中获取页面元素的方式。

    ```
    $('#div') == document.getElementById('div');
    ```
- 漂亮的页面动态效果

    - jQuery中内置了一系列的动画效果，如淡入淡出、元素移除等动态特效。
- 创建Ajax无刷新网页
    - 通过Ajax，可以对页面进行局部刷新，而不必每次数据更新都需要刷新整个页面。
- 对JavaScript的增强
    - jQuery提供了对基本JavaScript结构的增强，如元素迭代、数组处理等操作。
- 增强的事件处理
    - jQuery提供了各种页面事件，避免在HTML中添加太多的事件处理代码。
- 更改网页内容
    - jQuery可以修改网页中的内容，简化了原本使用JavaScript处理的方式，如更改网页文本、插入图像等。

如何使用jQuery

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>隐藏标签</title>
    <!-- 引入外部jquery文件-->
    <script src="jquery-3.2.1.min.js">
</script>
</head>
```

```
 9 <body>
10 <p>这是一个p标签</p>
11 <button id="hidden">隐藏p标签</button>
12 </body>
13 <script>
14     $('#hidden').click(function () {
15         //$('p').hide();//隐藏
16         //$('p').show();//显示
17         $('p').toggle();//显示/隐藏来回切
   换
18     });
19 </script>
20 </html>
```

jQuery选择器

基本选择器

ID选择器

使用公式： $("#id")

使用示例： $("#box") //获取id属性为box的元素

介绍：ID选择器#id就是利用DOM元素的id属性值来筛选匹配的元素，并以iQuery包装集的形式返回给对象。

元素选择器

使用公式：$（"element"）

使用示例：$（“div”） //获取所有的元素

介绍：元素选择器是根据元素名称匹配相应的元素。元素选择器指向的是DOM元素的标记名，也就是说元素选择器是根据元素的标记名选择的。

类名选择器

使用公式：$（".class"）

使用示例：$（".box"） //获取class属性值为box的所有元素

介绍：类选择器是通过元素拥有的CSS类的名称查找匹配的DOM元素。在一个页面中，一个元素可以有多个CSS类，一个CSS类又可以匹配多个元素，如果有元素中有一个匹配类的名称就可以被类选择器选取到。简单地说类名选择器就是以元素具有的CSS类名称查找匹配的元素。

复合选择器

使用公式：$("selector1,selector2,.....,selectorN")

使用示例：$("div,#btn") //要查询文档中的全部的`<div>`元素和id属性为btn的元素

selector1:一个有效的选择器，可以是ID选择器、元素选择器或类名选择器等

selector2:另一个有效的选择器，可以是ID选择器、元素选择器或类名选择器等

selectorN:（可选择）任意多个选择器，可以是ID选择器、元素选择器或类名选择器等

介绍： 复合选择器将多个选择器（可以是ID选择器、元素选择器或是类名选择器）组合在一起，两个选择器之间以逗号","分隔，只要符合其中的任何一个筛选条件就会被匹配，返回的是一个集合形式的jQuery包装集，利用jQuery索引器可以取得集合中的jQuery对象。

注意：多种匹配条件的选择器并不是匹配同时满足这几个选择器的匹配条件的元素，而是将每个匹配的元素合并后一起返回。

通配符选择器

$（" * "） //取得页面上所有的DOM元素集合的jQuery包装集

层次选择器

ancestor descendant选择器

使用公式：$（"ancestor descendant"）

使用示例：$（"ul li"） //匹配ul元素下的全部li元素

ancestor是指任何有效的选择器。

descendant是用以匹配元素的选择器，并且ancestor所指定元素后代元素。

ancestor descendant选择器中ancestor代表祖先，descendant代表子孙，用于在给定的祖先元素下匹配所有的后代元素

parent>child选择器

使用公式：$（" parent>child"）

使用示例：$（" form>input"） //匹配表单中所有的子元素input

parent是指任何有效的选择器

child使用以匹配元素的选择器，并且它是parent元素的子元素

介绍：parent>child选择器中的parent代表父元素，child代表子元素，用于在给定的父元素下匹配所有的子元素，使用该选择器只能选择器父元素的子元素

prev+next选择器

使用公式：$（" prev+next"）

使用示例：$（"div+img"） // 匹配`<div>`标签后的`<img>`标记

prev是指任何有效的选择器

next是一个有效选择器并紧接着prev选择器

介绍：pev+next选择器用于匹配所有紧接在prev元素后的next元素，其中，prev和next是两个相同级别的元素

prev~siblings选择器

使用公式：$（"prev~siblings"）

prev是指任何有效的选择器

介绍：prev~siblings选择器用于匹配prev元素之后的所有siblings元素，其中prev和siblings是个相同辈元素

过滤选择器

简单过滤器

简单过滤器是指冒号开头的，通常用于实现简单过滤效果的过滤器

:frist

说明：匹配找到第一个元素，它是与选择器结合使用的

示例：$(" tr:frist ") //匹配表格的第一行

:last

说明：匹配找到最后一个元素，它是与选择器结合使用的

示例：$(" tr:last ") //匹配表格的最后一行

:even

说明：匹配所有索引值为偶数的元素，索引值从0开始计算

示例：$(" tr:even") //匹配索引为偶数一行

:odd

说明：匹配所有索引值为奇数的元素，索引值从0开始计数

示例：$("tr:odd") //匹配索引值为奇数的行

:eq(index)

说明：匹配一个给定索引值的元素

示例：$("div:eq(1)") //匹配第二个div元素

:gt(index)

说明：匹配所有大于给定索引值的元素

示例：$("span:gt(0)") //匹配索引大于0的span元素（注：大于0,而不包括0)

:lt(index)

说明：匹配所有小于给定索引值的元素

示例：$("div:lt(2)") //匹配索引小于2的div元素（注：小于2,而不包括2)

:header

说明：匹配h1,h2,h3.....之类的标题元素

示例：$(".cls:header") //匹配全部类名为cls的标题元素，如果":"前不写则匹配所有的标题元素

:not(selector)

说明：去除所有给定选择器匹配的元素

示例：$("input:not(:checked)") //匹配没有被选中的input元素

:animated

说明：匹配所有正在执行动画效果的元素

示例：$("div:animated") //匹配正在执行的动画的div元素

内容过滤器

内容过滤器就是通过DOM元素包含的文本内容以及是否含有匹配的元素进行筛选

:contains(text)

说明：匹配包含给定文本的元素

示例：$("li:contains('word')") //匹配含有"word"文本内容的元素

:empty

说明：匹配所有不包含子元素或者文本的空元素

示例：$("td:empty") //匹配不包含子元素或者文本的单元格

:has(selector)

说明：匹配含有选择器所匹配元素的元素

示例：$("td:has(p)") //匹配表格的单元格中还有

标记的单元格

:parent

说明：匹配含有子元素或者文本的元素

示例：$("td:parent") //匹配不为空的单元格，即在该单元格中还包括子元素或则文本

可见性过滤器

元素的可见状态有两种，分别是隐藏状态和显示状态。可见性过滤器就是利用元素的可见状态匹配元素的

:visible

说明：匹配所有可见元素

:hidden

说明：匹配所有不可见元素

注意：在应用:hidden过滤器时，display属性是none以及input元素的type属性为hidden的元素都会被匹配识别

表单对象的属性过滤器

表单对象的属性过滤器通过表单元素的状态属性（例如：选中、不可用等状态）匹配元素

:checked

说明：匹配所有选中的被选中元素

示例：$("input:checked") //匹配所有被选中的input元素

:disabled

说明：匹配所有不可用元素

示例：$("input:disenabled") //匹配所有不可用input元素

:enabled

说明：匹配所有可用的元素

示例：$("input:enabled") //匹配所有可用的input元素

:selected

说明：匹配所有选中的option元素

示例：$("select option:selected") //匹配所有被选中的选项元素

子元素过滤器

子元素选择器就是筛选给定某个元素的子元素，具体的过滤条件由选择器的种类而定

:first-child

说明：匹配所有给定元素的第一个子元素

示例：$("ul li:first-child") //匹配ul元素中的第一个子元素li

:last-child

说明：匹配所有给定元素的最后一个子元素

示例：$("ul li:last-child") //匹配ul元素中的最后一个子元素li

:only-child

说明：如果某个元素是它父元素中唯一的子元素，那么将会被匹配，如果父元素中含有其他元素，则不会被匹配

示例：$("ul li:only-child") //匹配只含有一个li元素的ul元素中的li

:nth-child(index/even/odd/equation)

说明：匹配可每个父元素下的第index个子或奇偶元素，index从1开始，而不是从0开始

示例：$("ul li :nth-child(even)") //匹配ul中索引值为偶数的li元素

$("ul li:nth-child(3)") //匹配ul中第3个li元素

属性选择器

属性选择器就是通过元素的属性作为过滤条件进行筛选对象

[attribute]

说明：匹配包含给定属性的元素

示例：$("div[name]") //匹配包含有name属性的div元素

[attribute=value]

说明：匹配属性值为value的元素

示例：$("div[name='test']") //匹配name属性是test的div元素

[attribute!=value]

说明：匹配属性值不等于value的元素

示例：$("div[name!='test']") //匹配name属性不是test的div元素

[attribute*=value]

说明：匹配属性值含有value的元素

示例：$("div[name*="test"]") //匹配name属性值中含有test值的div元素

[attribute^=value]

说明：匹配属性值以value开始的元素

示例：$("div[name^='test']") //匹配name属性以test开头的div元素

[attribute$=value]

说明：匹配属性值以value结束的元素

示例：$("div[name$='test']") //匹配name属性以test结尾的div元素

[selector1][selector2][selectorN]

说明：复合属性选择器，需要同时满足多个条件时使用

示例：$("div[id][name^='test']") //匹配具有id属性并且name属性是以test开头的div元素

表单选择器

表单选择器是匹配经常在表单内出现的元素。但是匹配的元素不一定在表单中

:input

说明：匹配所有的input元素

示例：

$(":input") //匹配所有的input元素

$("form:input") //匹配

标记中的所有input元素，需要注意，在form和冒号之间有一个空格

:button

说明：匹配所有的普通按钮，即type="button"的input元素

示例：$(":button") //匹配所有普通按钮

:checkbox

说明：匹配所有的复选框

示例：$(":checkbox") //匹配所有的复选框

:file

说明：匹配所有的文件域

示例：$(":file") //匹配所有的文件域

:hidden

说明：匹配所有的不可见元素，或者type为hidden的元素

示例：$(":hidden") //匹配所有的隐藏域

:image

说明：匹配所有的图像域

示例：$(":image") //匹配所有的图像域

:password

说明：匹配所有的密码域

示例：$(":password") //匹配所有的密码域

:radio

说明：匹配所有的单选按钮

示例：$(":radio") //匹配所有的单选按钮

:reset

说明：匹配所有的重置按钮，即type="reset"的input元素

示例：$(":reset") //匹配所有的重置按钮

:submit

说明：匹配所有的提交按钮，即type="submit"的input元素

示例：$(":submit") //匹配所有的提交按钮

:text

说明：匹配所有的单行文本框

示例：$(".text") //匹配所有的单行文本框

选择器中注意事项

选择器中含有特殊符号的注意事项

含有"."、"#"、"{"、"}"等特殊字符：根据W3C规定，属性值中是不能包含这些特殊字符的，但在实际的项目中偶尔会遇到这种表达式中含有"#"和"}"等特殊字符的情况。这时，如果按照普通方法去处理的话就会出现错误，解决这类错误的方法是使用转义符号将其转义。

```
1 <div id="li#db">liaidb</div>
2 <div id="lidb(1)">lilovedb</div>
```

如果按照普通方式来获取，例如：

```
$("#li#db");
$("#lilovedb(1)");
```

这样是不能正确获取到元素的，正确的写法如下：

```
$("#li\\#db");
$("#lilovedb\\(1\\)");
```

选择器中空格的注意事项

在实际应用当中，选择器中含有空格也是不容忽视的，多一个空格或则少一个空格也会得到截然不同的结果。

```
 <div class="name">
        <div style="display: none;">小李</div>
        <div style="display: none;">小王</div>
        <div style="display: none;">小明</div>
        <div style="display: none;" class="name">小张</div>
    </div>
    <div style="display: none;" class="name">小玉</div>
    <div style="display: none;" class="name">小刘</div>
```

使用如下的jQuery选择器分别获取它们

```
<script type="text/javascript">
    var $a=$(".name :hidden");
    var $b=$(".name:hidden");
    console.log($a)
    console.log($b)
</script>
```

jQuery核心函数

$(callback)

当DOM加载完成后，执行其中的函数。也就是jQuery的入口函数

```
$(function(){

});
```

```
$(document).ready(function(){

});
```

jQuery对象

基本函数

length

获取DOM元素个数

[index]/get(index)

得到对应位置的DOM元素

each()

遍历包含的所有DOM元素

```
$('div').each(function (e,a) {
                    console.log(a)
                })
```

index()

得到在所在兄弟元素中的下标

操作属性

操作所有属性

添加/获取

```
attr(name|properties|key,value|fn)
```

移除

```
removeAttr(name)
```

操作class属性

添加

```
addClass(class|fn)
```

移除

```
removeClass([class|fn])
```

切换

```
toggleClass(class|fn[,sw])
```

CSS样式

```
css(name|pro)
```

- css(styleName): 根据样式名得到对应的值
- css(styleName, value): 设置一个样式
- css({多个样式对}): 设置多个样式,样式值和value之间用:，样式之间用,隔开

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>获取/添加CSS样式</title>
    <!-- 引入外部jquery文件-->
    <script src="jquery-3.2.1.min.js">
</script>
```

```
 8      <style>
 9          .myClass{
10              width: 100px;
11              height: 100px;
12              background-color: #ff0000;
13          }
14      </style>
15  </head>
16  <body>
17  <div class="myClass" id="myId">
18  这个是一个div
19  </div>
20
21  <button id='add'>获取CSS样式</button>
22  <button id='modify'>修改CSS样式</button>
23  </body>
24  <script>
25      //jQuery的入口函数
26      $(function () {
27          $('#add').click(function () {
28              //获取div的背景颜色
29
     console.log($('#myId').css('background-
    color'));
30          });
31          $('#modify').click(function () {
```

```
32              $('#myId').css({'font-size':
    '12px','padding-top': '20px'})
33          });
34      })
35 </script>
36 </html>
```

HTML标签

获取/设置html标签

```
1 html([val|fn])
```

文本

结果是由所有匹配元素包含的文本内容组合起来的文本。这个方法对HTML和XML文档都有效

```
1 text([val|fn])
```

jQuery 文档处理

内部插入

- **append**：向当前匹配的所有元素内部最后面追加内容mo'we
- **appendTo**：把所有匹配的元素追加到另一个指定的元素元素集合中。
- **prepend**：向当前匹配的所有元素内部最前面追加内容
- **prependTo**：把所有匹配的元素前置到另一个、指定的元素元素集合中

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>jQuery 文档处理--内部插入</title>
    <!-- 引入外部jquery文件-->
    <script src="jquery-3.2.1.min.js"></script>
    <style>
        .myClass{
            width: 400px;
            height: 400px;
            background-color: aliceblue;
        }

```

```
    </style>
</head>
<body>
<img src="images/home.png">
<img src="images/img.png">
<img src="images/my.png">
<img src="images/search.png">
<div class="myClass" id="myId">
    <p>这是一个段落标签</p>
</div>

<button id='append'>append</button>
<button id='append_to'>appendTo</button>
<button id='prepend'>prepend</button>
<button
id='prepend_to'>prependTo</button>
</body>
<script>
    //jQuery的入口函数
    $(function () {
        $('#append').click(function () {
            $('#myId').append('<span>你
今天在干嘛呢？</span>')
        });

        $('#append_to').click(function
() {
```

```
39              console.log( $('img'));
40
    $('img').eq(3).appendTo($('#myId'));
41          });
42
43          $('#prepend').click(function ()
    {
44              $('#myId').prepend('<span>你
    今天在干嘛呢？</span>')
45          });
46
47          $('#prepend_to').click(function
    () {
48
    $('img').eq(2).prependTo($('#myId'));
49          })
50      });
51
52  </script>
53  </html>
```

外部插入

after:在每个匹配的元素之后插入内容。

before:在每个匹配的元素之前插入内容

insertAfter:把所有匹配的元素插入到另一个指定的元素元素集合的后面。

insertBefore:把所有匹配的元素插入到另一个指定的元素元素集合的前面。

```
A.after(B);把B作为A的兄弟节点插入到A的后面，作为A的下一个兄弟节点
A.before(B);把B作为A的兄弟节点插入到A的前面，作为A的上一个兄弟节点
insertAfter，insertBefore把前面作为兄弟节点插入到后面的元素的后面和前面
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>jQuery 文档处理--外部插入</title>
    <!-- 引入外部jquery文件-->
    <script src="jquery-3.2.1.min.js"></script>
    <style>
        .myClass{
            width: 400px;
            height: 400px;
```

```
            background-color: aliceblue;
        }

    </style>
</head>
<body>
<img src="images/home.png">
<img src="images/img.png">
<img src="images/my.png">
<img src="images/search.png">
<div class="myClass" id="myId">
    <p>这是一个段落标签</p>
</div>

<button id='before'>before</button>
<button id='after'>after</button>
<button id='insertBefore'>insertBefore</button>
<button id='insertAfter'>insertAfter</button>
</body>
<script>
    //jQuery的入口函数
    $(function () {
        $('#before').click(function () {
            $('#myId').before('<div><img src="images/search.png"></div>')
```

```
36          });
37
38          $('#after').click(function () {
39              $('#myId').after('<div><img
    src="images/search.png"></div>')
40          });
41
42
     $('#insertBefore').click(function () {
43
     $('img').eq(0).insertBefore($('#myId'))
    ;
44          });
45
46          $('#insertAfter').click(function
    () {
47
     $('img').eq(2).insertAfter($('#myId'));
48          })
49      });
50
51 </script>
52 </html>
```

包裹

wrap:把所有匹配的元素用其他元素的结构化标记包裹起来。

unwrap:这个方法将移出元素的父元素

wrapAll:将所有匹配的元素用单个元素包裹起来

wrapInner:将每一个匹配的元素的子内容(包括文本节点)用一个HTML结构包裹起来

> wrap()为每一个匹配的元素都包裹一次。这种包装对于在文档中插入额外的结构化标记最有用，而且它不会破坏原始文档的语义品质
>
> wrapAll()把整个文档匹配的元素放到一起用一个元素包裹起来，这个会破坏原始文档的结构

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>jQuery 文档处理--包裹</title>
    <!-- 引入外部jquery文件-->
    <script src="jquery-3.2.1.min.js">
</script>
    <style>
        .myClass{
            width: 400px;
            height: 400px;
```

```
            background-color: aliceblue;
        }

    </style>
</head>
<body>
<div>
    <p>Hello</p>
    <p>cruel</p>
    <button>按钮</button>
    <p>World</p>
</div>


<p>你好</p>
<button id='wrap'>wrap</button>
<button id='unwrap'>unwrap</button>
<button id='wrapAll'>wrapAll</button>
<button
id='warpinner'>warpinner</button>
<p>你好</p>
</body>
<script>
    //jQuery的入口函数
    $(function () {
        $('#wrap').click(function () {
            $('p').wrap('<div></div>');
```

```
38          });
39
40          $('#unwrap').click(function () {
41              $('p').unwrap();
42          });
43
44          $('#wrapAll').click(function ()
    {
45              $('p').wrapAll("<div>
    </div>")
46          });
47
48          $('#warpinner').click(function
    () {
49              $('p').wrapInner("<div>
    </div>")
50          })
51      });
52
53  </script>
54  </html>
```

替换

`replaceWith()：` 将所有匹配的元素替换成指定的HTML或DOM元素。

`replaceAll()：` 用匹配的元素替换掉所有 selector匹配到的元素

删除

`empty()：` 删除匹配的元素集合中所有的子节点。

`remove()：` 从DOM中删除所有匹配的元素。这个方法不会把匹配的元素从jQuery对象中删除，因而可以在将来再使用这些匹配的元素。但除了这个元素本身得以保留之外，其他的比如绑定的事件，附加的数据等都会被移除。

`detach()：` 从DOM中删除所有匹配的元素。这个方法不会把匹配的元素从jQuery对象中删除，因而可以在将来再使用这些匹配的元素。与remove()不同的是，所有绑定的事件、附加的数据等都会保留下来。

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>删除</title>
6     <script src="js/jquery-
  3.6.0.min.js"></script>
7 </head>
8 <body>
```

```
 9 <div style="background-color:
   #0C0C0C;margin-bottom: 40px">
10     <p id="p1" style="color:
   #ff0000;font-size: 34px">empty</p>
11 </div>
12 <div style="background-color:
   #0C0C0C;margin-bottom: 40px">
13     <p id="p2" style="color:
   #ff0000;font-size: 34px">remove</p>
14 </div>
15 <div style="background-color:
   #0C0C0C;margin-bottom: 40px">
16     <p id="p3" style="color:
   #ff0000;font-size: 34px">detach</p>
17 </div>
18
19 <button id="bu">找回被remove的元素
   </button>
20 <div id="div" style="background-color:
   #00FFFF;width: 300px;height: 300px">
   </div>
21 <script>
22     /**
23      * empty: 删除匹配的元素集合中所有的子节
   点。
24
25      remove: 从DOM中删除所有匹配的元素。
```

```
26          这个方法不会把匹配的元素从jQuery对象中删除，
27          因而可以在将来再使用这些匹配的元素。
28          但除了这个元素本身得以保留之外，
29          其他的比如绑定的事件，附加的数据等都会被移除。
30
31          detach:
32          从DOM中删除所有匹配的元素。
33          这个方法不会把匹配的元素从jQuery对象中删除，
34          因而可以在将来再使用这些匹配的元素。
35          与remove()不同的是，
36          所有绑定的事件、附加的数据等都会保留下来
37          */
38
39      $(function () {
40          $('#p1').click(function () {
41              $(this).parent().empty()
42          })
43          var dom = null;
44
45          $('#p2').click(function () {
46              dom = $(this).remove();
47          });
48
49          $('#bu').click(function () {
```

```
50              $('#div').append(dom);
51          });
52
53          $('#p3').click(function () {
54              dom = $(this).detach()
55          });
56      });
57  </script>
58  </body>
59  </html>
```

复制

`clone()`: 克隆匹配的DOM元素并且选中这些克隆的副本。在想把DOM文档中元素的副本添加到其他位置时这个函数非常有用。

过滤

`eq`: 获取当前链式操作中第N个jQuery对象，返回jQuery对象，当参数大于等于0时为正向选取，比如0代表第一个，1代表第二个。当参数为负数时为反向选取，比如-1为倒数第一个，具体可以看以下示例。类似

的有get(index),不过get(index)返回的是DOM对象。

`first:` 获取第一个元素

`last:`获取最后个元素

`hasClass:` 检查当前的元素是否含有某个特定的类，如果有，则返回true。

`filter:`筛选出与指定表达式匹配的元素集合。这个方法用于缩小匹配的范围。用逗号分隔多个表达式

`is:`根据选择器、DOM元素或 jQuery 对象来检测匹配元素集合，如果其中至少有一个元素符合这个给定的表达式就返回true。如果没有元素符合，或者表达式无效，都返回'false'。 '''注意：'''在jQuery 1.3中才对所有表达式提供了支持。在先前版本中，如果提供了复杂的表达式，比如层级选择器（比如 +，~ 和 > ），始终会返回true

`slice:`选取一个匹配的子集与原来的slice方法类似

查找

`children:`

`closest:`

`find:`

next:

nextAll:

nextUnit:

offsetParent:

parent:

parents:

prev:

prevAll:

prevUntil:

siblings:

jQuery事件

jQuery动画效果