# jQuery框架

# 目录

[jQuery框架 1](#_Toc22005)

[目录 2](#_Toc14956)

[一、复习上次课的内容 3](#_Toc27908)

[二、序与迭代 6](#_Toc22260)

[2.1 eq()方法 6](#_Toc28391)

[2.2 index()方法 7](#_Toc21751)

[2.3 each() 7](#_Toc19322)

[2.4 size()方法和length属性 8](#_Toc5770)

[2.5 get()方法 8](#_Toc28544)

[三、动画相关方法 9](#_Toc9308)

[3.1 内置show()、hide()、toggle()方法 9](#_Toc15357)

[3.2 slideDown()、slideUp()、slideToggle()方法 10](#_Toc12846)

[3.3 fadeIn()、fadeOut()、fadeTo()、fadeToggle()方法 11](#_Toc14672)

[3.4 stop() 11](#_Toc17400)

[3.5 finish() 12](#_Toc27716)

[3.6 delay() 12](#_Toc24907)

[3.7 is(":animated") 12](#_Toc9318)

[四、节点关系 14](#_Toc28849)

[4.1 原生JS中nodeType属性 14](#_Toc5740)

[4.2 原生JS中的节点关系-childNodes 15](#_Toc25367)

[4.3 原生JS中的节点关系-parentNode 16](#_Toc27248)

[4.4 previousSibling、nextSibling 16](#_Toc22409)

# 一、复习上次课的内容

先解决大家一个疑惑：工作的时候用原生JavaScript写效果，还是jQuery？

这是一个伪命题，原生JS开发效果，我们也会用一些轮子简化编程。我们关心的永远是上层业务，而不是底层的一些苟且（比如浏览器兼容、获得计算样式啥的）。所以，你会发现，即使用原生JS写效果，也编写了fetchComputedStyle()、animate()函数，已经相当于把JS的纯底层的复杂性“屏蔽”掉了。所以公司如果因为各种理由不用jQuery（比如觉得jQuery尺寸大，影响页面速度），那么一定也会使用JS轮子，或者其他类似jQuery的东西。你会发现jQuery没有为我们简化业务，轮播图该写逻辑还是写逻辑，只不过一些事情方便了。所以，赶紧好好学习jQuery，好好学习原生，好好学习逻辑，好好学习具体业务。

● $函数

|  |
| --- |
| 1. $("#box").css("background-color","red"); |

一定要加引号，只有

|  |
| --- |
| 1. $(this) 2. $(document) 3. $(window) |

没有引号。

$(“#box”)是jQuery对象，而不是JS原生对象。加[0]就能转为原生对象

|  |
| --- |
| 1. $("#box")[0].style.backgroundColor = "red"; |

这里我们补充一点，jQuery选择的如果是很多元素，那么[0]转化的是一个元素。[1]、[2]、[3]……

|  |
| --- |
| 1. $("#box")[0].style.backgroundColor = "red"; 2. $("#box")[1].style.backgroundColor = "red"; 3. $("#box")[2].style.backgroundColor = "red"; |

$函数和jQuery是同名函数，

|  |
| --- |
| 1. jQuery("#box"); |

● 选择器和筛选器

选择器css2.1支持，也支持部分css3的选择器。

筛选器，写在引号里面，用:当做筛选的功能符。

|  |
| --- |
| 1. $("div:first") 2. $("div:last") 3. $("div:eq(0)") 4. $("div:lt(4)") 5. $("div:gt(3)") 6. $("div:odd") 7. $("div:even") |

特别的eq能单独提炼为方法：

|  |
| --- |
| 1. $("div").eq(3).css("background","red"); |

● css()方法

css函数用来读取、设置元素的css样式。可以读取计算后的样式的。

读取的时候一个参数，就是k

|  |
| --- |
| 1. $("div:eq(5)").css("width") |

设置的时候：

|  |
| --- |
| 1. $("div:eq(2)").css("background-color"**,**"red"); 2. $("div:eq(2)").css("width"**,**"40px"); 3. $("div:eq(2)").css("height"**,**60); |

可以用JSON来简化：

|  |
| --- |
| 1. var json = { 2. "background-color" : "red", 3. "width" : 40, 4. "height" : 60 5. } 6. $("div:odd").css(json); |

特别的，可以有+=写法：

|  |
| --- |
| 1. $(this).css("width","+=10px"); |

● animate()方法

对象打点调用animate方法，就能让这个对象运动：

|  |
| --- |
| 1. $("div:odd").animate(json,1000,function(){ 2. $(this).css("background-color","red") 3. }); |

你要记住这个事情，background-color是不能渐变的。页面上如果想要使用background-color的过渡效果，慢慢从红色变为蓝色，必须使用css3。

jQuery中提供了非常牛逼的动画队列功能，相同元素的animate()方法会累积

|  |
| --- |
| 1. $("div").animate({"width":600},1000); 2. $("div").animate({"height":200},1000); |

不同元素的动画是同时进行的

|  |
| --- |
| 1. $("div:eq(0)").animate({"width":600},1000); 2. $("div:eq(1)").animate({"height":600},1000); |

● 监听：

|  |
| --- |
| 1. $("div:even").click(function(){ 2. $(this).animate({"width":600},2000); 3. }); |

监听此时没有on的

有一个高度来看jQuery，就是你一定要悟出来一个事儿：

jQuery中所有的东西，都是批量的。并且不用写for循环语句：

|  |
| --- |
| 1. $("div").css("background","red"); //设置样式是批量的 2. $("div").animate({"width":300},4000); //设置动画是批量的 3. $("div").click(function(){}); //设置事件是批量的 |

凭什么它这么牛逼。

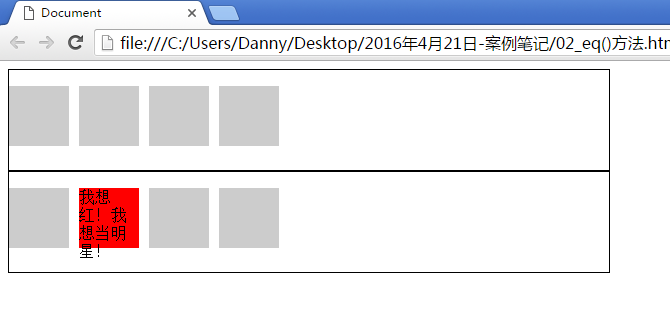
$()函数返回的是一组元素，每个方法比如css方法内部都有for循环迭代。说白了，不是没有for循环只不过for循环隐藏到css()方法里面了、隐藏到animate()方法里面了、隐藏到了click()方法里了。

# 二、序与迭代

## 2.1 eq()方法

来看下面的结构，两个div中各自有4个p，让其中一个p变红：

|  |
| --- |
| 1. <div class="box1"> 2. <p></p> 3. <p class="teshu"></p> 4. <p class="teshu"></p> 5. <p></p> 6. </div> 7. <div class="box2"> 8. <p class="teshu"></p> 9. <p class="teshu">我想红！我想当明星！</p> 10. <p></p> 11. <p></p> 12. </div> |



现在我们想选择box2里面的1号下标的p：

|  |
| --- |
| 1. $(".box2 p").eq(**1**) 2. $("p").eq(**5**) 3. $(".teshu").eq(**3**) 4. $(".box2 .teshu").eq(**1**) |

$()函数将返回一个对象队列，用eq来精确选择这个序列中的某个元素。到底eq几是这个元素呢？**仰赖$()的序列是什么**。

如同爸爸去哪儿，村长吹哨，把所有的儿子p都吹出来排队了，然后选择下标是第几的元素。

同样的，lt、gt、odd、even、first、last都是这个机理。

## 2.2 index()方法

返回这个元素在亲兄弟中的排名，**无视选择器怎么选**。

$(this).index()是一个很常见的写法，表示触发这个事件的元素，在自己亲兄弟中的排名：

|  |
| --- |
| 1. $(".teshu").click(function(){ 2. alert($(this).index()); 3. }) |

对应：

点击box1里面的p，让对应的box2里面的p变红：

|  |
| --- |
| 1. //事件监听要给box1中的所有p标签， 2. $(".box1 p").click(function(){ 3. //有变化的是box2中对应的p 4. $(".box2 p").eq($(this).index()).css("background-color","red"); 5. }); |

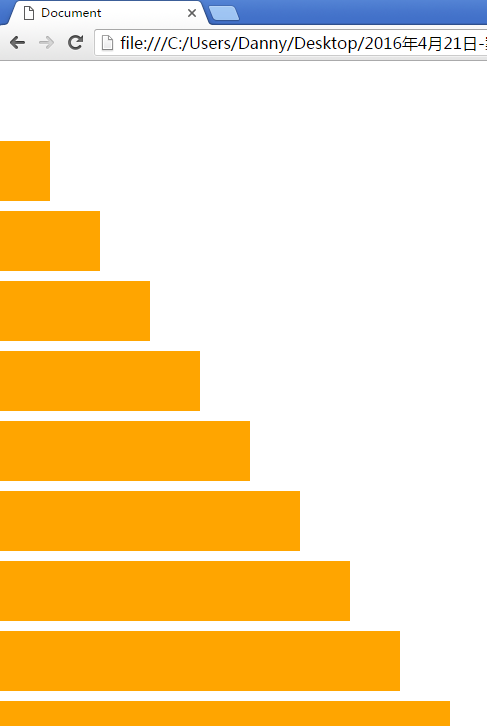
最后强调一下，是index()方法，而不是属性！

## 2.3 each()

each()表示遍历节点，也叫作迭代符合条件的节点。

each()语句就好比派出一个侦察兵，挨家挨户去敲门，敲开门之后做什么事情，写在function(){}里面，这里面的$(this)表示敲开门的这家。

|  |
| --- |
| 1. $("p").each(function(i){ 2. $(this).animate({"width":50 \* i},1000); 3. }); |



$(“p”)选择了页面上的所有的p，现在我们想分别为这个p设置不同的动画终点。那么each语句就很好用，会依次遍历所有的p，$(this)表示你现在正在遍历的p。

## 2.4 size()方法和length属性

jQuery 对象中元素的个数。

前面$()的元素页面上一共有几个，length、size()返回的都是同一个数值，就是个数。

|  |
| --- |
| 1. $("p").length 2. $("p").size() |

## 2.5 get()方法

get()方法和eq()方法基本一致，都仰赖$()的序列。

eq()返回的是jQuery对象，而get()返回的是原生JS对象。jQuery对象后面要跟着jQuery方法，原生对象后面要跟着原生属性、方法：

|  |
| --- |
| 1. $("p").**eq(2)**.html("哈哈哈哈哈哈"); |

等价于：

|  |
| --- |
| 1. $("p").**get(2)**.innerHTML = "哈哈哈哈哈哈"; |

等价于：

|  |
| --- |
| 1. $("p").**eq(2)[0]**.innerHTML = "哈哈哈哈哈哈"; |

# 三、动画相关方法

我们昨天学习了animate()方法，很好用，今天继续深入学习更多的动画相关方法。

## 3.1 内置show()、hide()、toggle()方法

show()显示、hide()隐藏、toggle()切换

|  |
| --- |
| 1. $("div").show(); //让一个本身是display:none;元素显示 2. $("div").hide(); //隐藏元素display:none; 3. $("div").toggle(); //切换显示状态。 4. //自行带有判断，如果可见，就隐藏；否则显示。 |

特别的，如果show()、hide()、toggle()里面有数值，将变为动画：

|  |
| --- |
| 1. $("div").show(1000); |

此时display:none;的元素，将从左上角徐徐展开。动画机理：

这个display:none;的元素会变为显示的，然后瞬间将宽度、高度、opacity设为0，然后徐徐展开。

甚至可以加回调函数：

|  |
| --- |
| 1. $("div").toggle(1000,function(){ 2. alert("土狗完毕！"); 3. }); |

原来如此！语法就是

|  |
| --- |
| 1. $("div").show([时间],[回调函数]); |

[]表示这个参数可选。

## 3.2 slideDown()、slideUp()、slideToggle()方法

slideDown : 下滑展开

slideUp：上滑收回

slideToggle : 滑动切换

|  |
| --- |
| 1. $("div").slideDown(); |

slideDown()的起点一定是display:none换句话说，只有display:none的元素，才能够调用slideDown()

动画机理：

一个display:none的元素，瞬间显示，瞬间高度变为0，然后jQuery自己捕捉原有的height设置为动画的终点。

等价于4条语句：

|  |
| --- |
| 1. $("div").show(); //瞬间显示 2. var oldHeight = $("div").css("height"); //记忆住原有的高度 3. $("div").css("height",0); //瞬间变为0 4. $("div").animate({"height" : oldHeight},1000); //动画！终点是oldHeight |

所以你会发现，jQuery真的很伟大！一个slideDown()封装了那么多逻辑！

相反的，slideUp()的终点就是display:none;

同样的，slideDown、slideUp、slideToggle里面可以写动画时间、回调函数。

我们这里讲解了一个案例：水平菜单。

注意水平菜单的html结构：

|  |
| --- |
| 1. <div class="nav"> 2. <ul> 3. <li><a href=""></a></li> 4. <li> 5. <a href=""></a> 6. <div class="dropdown"> 7. <div class="inner"> 9. </div> 10. </div> 11. </li> 12. <li><a href=""></a></li> 13. <li><a href=""></a></li> 14. <li><a href=""></a></li> 15. </ul> 16. </div> |

注意css：

.dropdown这个盒子，一定要绝对定位，top值必须和nav的高度一致，必须紧密贴合nav。

所以如果你想表现菜单和nav不贴合，那么必须套一个inner，用padding来表示缝隙。

js事件：

注意事件加给li，不要给a

## 3.3 fadeIn()、fadeOut()、fadeTo()、fadeToggle()方法

fadeIn()淡入

fadeOut()淡出

fadeTo() 淡到那个数

fadeToggle() 淡出入切换

fadeIn()的起点是display:none;换句话说，只有display:none的元素，才能执行fadeIn()

|  |
| --- |
| 1. $("div").fadeIn(5000); |

动画机理：

一个display:none的元素，瞬间可见，然后瞬间变为opacity:0，往自己的opacity上变。如果没有设置opacity，就往1变。

fadeTo有三个参数，第一个参数是动画的时间，第二个参数是要变到的透明度，第三个参数是回调函数。

|  |
| --- |
| 1. $("div").fadeTo(1000,0.3); |

fadeTo的起点不一定是display:none;

IE6、7、8兼容，不用关心filter这个东西了，jQuery已经帮你写了兼容。

## 3.4 stop()

stop()挺有意思的：

停止当前的animate动画，但是不清除队列，立即执行后面的animate动画：

|  |
| --- |
| 1. $("div").stop();  **//等价于$(“div”).stop(false,false);** |

停止当前的animate动画，并且清除队列，盒子留在了此时的位置：

|  |
| --- |
| 1. $("div").stop(true); **//等价于$(“div”).stop(true,false);** |

瞬间完成当前的animate动画，并且清除队列：

|  |
| --- |
| 1. $("div").stop(true,true); |

瞬间完成当前的animate动画，但是不清楚队列，立即执行后面的动画：

|  |
| --- |
| 1. $("div").stop(false,true); |

公式：

|  |
| --- |
| 1. stop(是否清除队列,是否瞬间完成当前动画) |

如果没有写true或者false，默认是false

## 3.5 finish()

finish()瞬间完成所有动画队列！

|  |
| --- |
| 1. $("div").finish(); |

stop可以用来防止动画的积累：

|  |
| --- |
| 1. //连续打点语法，先清除所有的动画队列，然后执行新的动画 2. $("div").stop(true).animate({"left":100},1000); |

## 3.6 delay()

delay延迟，可以使用连续打点，必须放在运动语句之前。

|  |
| --- |
| 1. $("div").delay(1000).animate({"left":500},1000); 2. $("div").delay(1000).slideUp(); 3. $("div").delay(1000).hide(**1**); //必须写1，写1了就是运动 |

小窍门，让所有的img元素，都延迟不同的时间入场：

|  |
| --- |
| 1. $("img").each(function(i){ 2. //attr就表示得到标签内的属性 3. $(this).delay(i \* 1000).fadeIn(1000); 4. }); |

## 3.7 is(":animated")

is()方法表示身份探测，返回true、false。

is表示“是不是”，而不是“是”

比如，判断点击的这个p是不是有t这个类：

|  |
| --- |
| 1. $("p").click(function(){ 2. alert( $(this).is(".t") ); 3. }); |

is里面可以写筛选器：

判断点击的这个p是不是序号是奇数：

|  |
| --- |
| 1. $(this).is("p:odd") |

判断点击的这个p是不是序号小于3：

|  |
| --- |
| 1. $(this).is("p:lt(3)") |

还可以写

|  |
| --- |
| 1. is(":animated") |

判断这个元素是否在运动中。

判断是否在运动中，可以防止动画的积累：

|  |
| --- |
| 1. if($(this).children(".dropbox").is(":animated")){ 2. return; 3. } |

只要动画正在进行，那么我不仅受更多的命令。

我们之前学习的stop()，哲学不一样，动画正在进行，立即停止手上的动画，执行新的命令。

# 四、节点关系

## 4.1 原生JS中nodeType属性

节点类型属性



任何的HTML元素，都有nodeType属性，值有1~11，老师为大家精简到了5个：

1： 元素节点

3： 文本节点

8： 注释节点

9： document节点

10: DTD

比如结构：

|  |
| --- |
| 1. <div id="box">文本<p></p></div> |

程序：

|  |
| --- |
| 1. var box = document.getElementById("box"); 2. alert(box.nodeType); //1 3. alert(box.childNodes[0].nodeType); //3 4. alert(box.childNodes[1].nodeType); //1 |

|  |
| --- |
| 1. alert(document.nodeType); //9 |

## 4.2 原生JS中的节点关系-childNodes

任何节点都有childNodes属性，是一个类数组对象，存放着所有自己的儿子。

**注意，这里有重大兼容性问题：**

结构

|  |
| --- |
| 1. <div id="box"> 2. <p></p> 3. </div> |

Chrome、IE9、IE10……高级浏览器，认为：

|  |
| --- |
| 1. box.childNodes[0].nodeType //3 |

高级浏览器认为box的大儿子是文本节点。当然是空文本。

IE6、7、8认为：

|  |
| --- |
| 1. box.childNodes[0].nodeType //1 |

IE6、7、8认为box的大儿子是p。

所以为了没有兼容问题，需要遍历节点的时候，HTML结构就不能有空格。

面试题：

|  |
| --- |
| 1. <div id="box"> 2. <p></p> 3. <p></p> 4. <p></p> 5. <p></p> 6. </div> |

|  |
| --- |
| 1. document.getElementById(‘box’).childNodes.length; //高级浏览器9，低级浏览器4 |

怎么解决这个差异呢？放弃原有的数组，重新遍历儿子数组，把所有nodeType为1的元素组成一个新的数组

|  |
| --- |
| 1. var childs = []; 2. for(var i = 0 ; i < box.childNodes.length ; i++){ 3. **if(box.childNodes[i].nodeType == 1){** 4. childs.push(box.childNodes[i]); 5. **}** 6. } 7. childs[1].style.background = "red"; |

firstChild属性、lastChild属性： 也不好用，IE6、7、8认为firstChild是节点， 而Chrome认为firstChild是空文本

## 4.3 原生JS中的节点关系-parentNode

注意childNodes儿子可以有很多 ，parendNode父亲只能有1个

|  |
| --- |
| 1. 某个元素.parentNode |

## 4.4 previousSibling、nextSibling

上一个同胞兄弟，下一个同胞兄弟。

需要注意的是，天大的浏览器兼容问题出现了：

|  |
| --- |
| 1. <div id="box"> 2. <p>AAA</p> 3. <p>BBB</p> 4. <p>CCC</p> 5. <p>DDD</p> 6. </div> |

|  |
| --- |
| 1. ps[2].previousSibling //低级浏览器就是BBB那个p，高级浏览器是空文本节点 |

让小明节点的之前的所有节点，都变红，原生JS挺难的：

|  |
| --- |
| 1. var prev = xiaoming; 2. **while(prev = prev.previousSibling){** 3. if(prev.nodeType == 1){ 4. prev.style.background = "red"; 5. } 6. **}** |