# jquery

# 目录

[jQuery框架 1](#_Toc10557)

[目录 2](#_Toc19180)

[一、传统轮播和呼吸轮播复习 3](#_Toc30525)

[1.1 传统轮播 3](#_Toc29920)

[1.2 三位置传统轮播图 3](#_Toc4131)

[1.3 呼吸轮播（交叉淡入淡出轮播） 4](#_Toc28164)

[二、异形滚动 5](#_Toc16599)

[三、jQuery框架 6](#_Toc20647)

[3.1 JavaScript编程比较恶心的地方 6](#_Toc24397)

[3.2 jQuery就是轮子轮子轮子轮子轮子的集合 6](#_Toc13387)

[3.3 简介 7](#_Toc16624)

[四、jQuery整体感知 8](#_Toc3289)

[4.1 选择问题轻松解决 8](#_Toc21893)

[4.2 样式问题轻松解决 8](#_Toc12406)

[4.3 动画的问题轻松解决 9](#_Toc4658)

[4.4 批量添加监听、节点关系 9](#_Toc8449)

[五、$()函数 10](#_Toc30076)

[5.1 jQuery对象不是原生JS对象 10](#_Toc24194)

[5.2 引号问题 10](#_Toc29823)

[5.3 支持的选择器 11](#_Toc29200)

[5.4 筛选器 11](#_Toc25266)

[六、CSS函数 12](#_Toc7423)

[七、animate函数 13](#_Toc12525)

[八、事件监听 14](#_Toc13903)

先不直接介绍jQuery，先把之前遗留的问题讲了

# 一、传统轮播和呼吸轮播复习

## 1.1 传统轮播

是一个a标签，但是我们不希望点击之后有任何默认的事情，默认事情通常是刷新当前页：

|  |
| --- |
| 1. <a href="javascript:;" class="leftBtn" id="leftBtn"></a> |

最最关键的就是右按钮、左按钮的策略：

右按钮，处理最后一张的时候，先拉后瞬移：

|  |
| --- |
| 1. rightBtn.onclick = function(){ 2. idx ++; 3. animate(m\_unit, {.......},500,function(){ 4. if(idx > 4){ 5. idx = 0; 6. m\_unit.style.left = “0px”; 7. } 8. }) 9. } |

左按钮，处理第一张的时候，先瞬移后拉动：

|  |
| --- |
| 1. leftBtn.onclick = function(){ 2. idx --; 3. if(idx < 0 ){ 4. idx = 4; 5. m\_unit.style.left = -560 \* 5 + "px"; 6. } 7. animate(m\_unit, {.....},500) 8. } |

## 1.2 三位置传统轮播图

页面效果，随着人们的认识深入，就产生更多的原理。

传统轮播的“火车法”的问题是，小圆点点击的时候，拉动的图片，跳跃性大。

比如现在是0号图，点击3号小圆点，就会看见1号、2号，停留在3号图。不是平滑过渡。

三位置传统轮播图，特点是，所有的li不是火车，而是都绝对定位。他们的left值只有三种可能：

-560 0 560



## 1.3 呼吸轮播（交叉淡入淡出轮播）

摞在一起，不管点击哪个按钮，老图淡出，新图淡入。点击小圆点，老图淡出，新图淡入。

我们讲一个bug，就是元素opacity:0了，也能够被点击。

解决办法就是，页面上有且又有一个li的display:block；

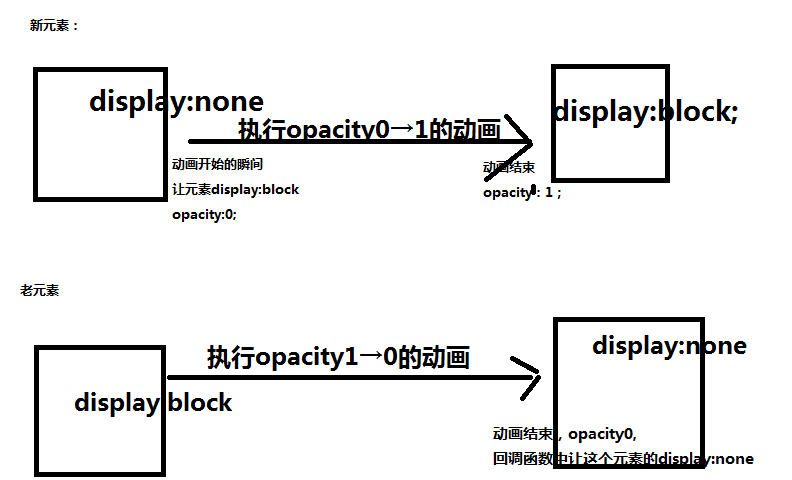
当点击右按钮的时候：

当前这个图片执行opacity1→0的动画，动画结束之后，让这个元素display:none;

新图片执行opacity0→1的动画，动画开始之前，瞬间让新图片display:block;然后再动画

一定要牢记：呼吸轮播动画的起点是display:block； 动画的终点是display:none;

但是JS是没法过渡display属性的，所以我们就往中间加上了opacity



# 二、异形滚动

每次老师吹哨的时候：

小明擦玻璃

小红拖地

小刚扫地

让小明改名叫小红

让小红改名叫小刚

让小刚改名叫小明

所以老师每次吹哨，这三个学生依次做不同的劳动。

先运动，然后改变身份。

lisArr = [lis.stateA,lis.stateB,lis.stateC,lis.stateD……]

点击一下按钮，我们永远让数组的第0项瞬移到6号位置，第1项动画变为0，第2项动画变为1……

lisArr.push(lisArr.shifi());

[lis.stateB,lis.stateC,lis.stateD……lis.stateA]

# 三、jQuery框架

## 3.1 JavaScript编程比较恶心的地方

恶心1：选择元素麻烦，全线兼容的方法只有getElementById()和getElementsByTagName()两个。其他的方法是不都兼容的。getElementsByClassName()通过类名选择元素，IE9开始兼容。

恶心2：样式操作麻烦，得到原生样式，需要我们自己造轮子fetchComputedStyle()

恶心3：动画麻烦，需要我们自己造轮子animate();

恶心4：批量控制麻烦，大量出现的for循环语句；排他操作麻烦

恶心5：HTML节点操作麻烦

恶心的东西，能用“轮子”来解决，事实上我们已经造了两个轮子：fetchComputedStyle、animate。

能不能把所有轮子组合起来，成为一个框架呢？jQuery就是这样的一个东西。

## 3.2 jQuery就是轮子轮子轮子轮子轮子的集合

jQuery是DOM编程领域的霸主，极大的简化了原生JavaScript的DOM编程。

jQuery中含有丰富的轮子，完美解决了选择元素难、样式难、动画难、批量操作难等各种兼容问题，**让工程师只需要思考业务，而不必思考细枝末节的兼容问题**。

John Resig jQuery的第一创始人



## 3.3 简介

**官网：**www.jquery.com



**口号：**写更少的代码，做更多的事情

**官方的自我介绍：**

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. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

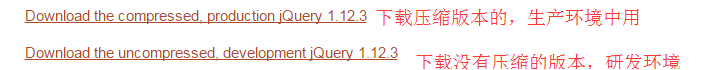
简单翻译：jQuery是一个快速、小型的、特性很多的JS库，它把很多事儿都变得简单。jQuery是免费的、开源的。

**jQuery有两条版本线**：1.X、2.X。

jQuery 2.x has the same API as jQuery 1.x, but does not support Internet Explorer 6, 7, or 8.

jQuery2.X和1.X的功能完全一致，API完全一致，但是2.X不兼容IE6、7、8。

下载的时候有两个选择：



# 四、jQuery整体感知

## 4.1 选择问题轻松解决

$()函数就是jQuery的核心函数，query就是选择的意思，也就是说jQuery的核心招牌功能就是选择元素：

|  |
| --- |
| 1. **$**("#box ul li.haha span").css("background-color","red"); |

语法：

|  |
| --- |
| 1. $(“选择器”) |

选择的就是所有符合条件的元素，而不是一个。

$可以用jQuery来代替，$和jQuery是同一个函数：

|  |
| --- |
| 1. **jQuery**("#box ul li.haha span").css("background-color","red"); |

一辈子就和document.getElmentById()无缘了，拜拜了你内！！

jQuery选择的元素，所有浏览器兼容！

## 4.2 样式问题轻松解决

得到样式（得到计算后的样式）：

|  |
| --- |
| 1. $(".box").css("width"); |

通过$()函数选择出来的东西，都是jQuery对象，所有的jQuery对象，都可以继续打点调用css函数，css函数已经封装了计算后的样式。

设置样式：

|  |
| --- |
| 1. $(".box").css({ 2. "width":700, 3. "height" : 300, 4. "opacity" : 0.4 5. }); |

语法

|  |
| --- |
| 1. $(“选择器”).css(JSON); |

## 4.3 动画的问题轻松解决

jQuery内部含有一个运动框架，比我们上课写的牛逼十万倍！

|  |
| --- |
| 1. $(".box").animate({"left":900},4000,function(){ 2. alert("运动完成"); 3. }); |

## 4.4 批量添加监听、节点关系

所有的li都有监听了：

|  |
| --- |
| 1. $(".circles ol li").mouseenter(function(){ 2. //自己变红，自己的兄弟恢复为橙色   $(this).css("background-color","red").siblings().css("background-color","orange");   1. }); |

siblings()表示兄弟节点。

# 五、$()函数

## 5.1 jQuery对象不是原生JS对象

$()函数，是招牌功能，能够根据CSS选择元素。

比如：

|  |
| --- |
| 1. $("#box") |

选择页面上id为box的盒子。

**注意，选择出来的东西，是一个类数组对象，是jQuery自己的对象，这个jQuery对象后面不能跟着原生JS的语法：**

|  |
| --- |
| 1. ~~$("#box").style.backgroundColor = "red";~~ |

因为.style.backgroundColor是原生JS语法，$()原则的对象是jQuery对象，不能跟着原生。

所以，如果想把jQuery对象，转为原生JS对象，要加[0]就行了：

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

$等价于jQuery

|  |
| --- |
| 1. jQuery("#box") 2. 等价于 3. $("#box") |

## 5.2 引号问题

|  |
| --- |
| 1. $("选择器") |

注意引号不能丢！！在jQuery世界中，只有三个东西不能加引号，其他必须加引号：

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

上述的三个东西，不能有引号，干吗用的，后面说。

## 5.3 支持的选择器

jQuery支持所有CSS2.1的选择器：

|  |
| --- |
| 1. $("p") 2. $(".box") 3. $("#box") 4. $("#box ul li") 5. $("li.special") 6. $("ol , ul") 7. $("\*") |

也支持部分CSS3的选择器，我们CSS3的选择器在CSS3课程介绍，所以也不讲了。

## 5.4 筛选器

这些都是关于序号的：

|  |
| --- |
| 1. $("p") 所有的p 2. $("p:first") 第一个p 3. $("p:last") 最后一个p 4. $("p:eq(3)") 下标为3的p 5. $("p:lt(3)") 下标小于3的p 6. $("p:gt(3)") 下标大于3的p 7. $("p:odd") 下标是奇数的p 8. $("p:even") 下标是偶数的p |

特别的，eq可以单独提炼为方法，可以连续打点：

|  |
| --- |
| 1. $("p").eq(3).animate({"width":400},1000); |

等价于

|  |
| --- |
| 1. $("p:eq(3)").animate({"width":400},1000); |

提炼出来的好处是，可以用变量

|  |
| --- |
| 1. var a = 3; 2. $("p").eq(a).animate({"width":400},1000); |

# 六、CSS函数

css函数可以读样式，可以设样式。

读样式，可以读取计算后样式，写一个参数，是不是驼峰，无所谓，但是必须加引号：

|  |
| --- |
| 1. $("p:first").css("background-color"); 2. $("p:first").css("backgroundColor"); |

设置样式，设置样式，有两种语法，**如果你只想设置一个样式，逗号隔开k和v**：

|  |
| --- |
| 1. $("p:odd").css("backgroundColor"**,**"blue"); |

**如果想设置很多样式，就写JSON:**

|  |
| --- |
| 1. $("p:odd").css(**JSON**); |

所有的数值，不需要单位：

|  |
| --- |
| 1. $("p:lt(4)").css({ 2. "width" : 20, 3. "height" : 20, 4. "backgroundColor" : "red" 5. }) |

当然，你也可以这样，不厌其烦的用逗号：

|  |
| --- |
| 1. $("p:lt(4)").css("width",20); 2. $("p:lt(4)").css("height",20); 3. $("p:lt(4)").css("backgroundColor","red"); |

特别的，还支持+=写法：

|  |
| --- |
| 1. $("p:eq(5)").css("width","+=20px"); |

# 七、animate函数

动画

|  |
| --- |
| 1. $("p").animate({"left":1000},2000,function(){ 2. $(this).css("background-color","red"); 3. }); |

我们骄傲的告诉大家，我们封装的animate语法基本和jQuery一样，只不过jQuery是对象打点：

|  |
| --- |
| 1. $("选择器").animate(终点JSON,动画时间,回调函数); |

有没有缓冲呢，有，jQuery需要插件来完成，我们日后说。

jQuery默认不是匀速，是easeInOut

和我们封装的框架不一样，jQuery默认有一个处理机制，叫做动画排队。当一个元素接收到了两个animate命令之后，后面的animate会排队：

|  |
| --- |
| 1. $("p").animate({"left":1000},2000); 2. $("p").animate({"top":400},2000); |

先2000毫秒横着跑，然后2000毫秒竖着跑。动画总时长4000。

如果想让元素斜着跑，就是同时变化left和top，就写在同一个JSON里面：

|  |
| --- |
| 1. $("p").animate({"left":1000,"top":400},2000); |

不同的元素，不排队，是同时的。

# 八、事件监听

jQuery颠覆了我们的行文习惯：

|  |
| --- |
| 1. $(".box1").click(function(){ 2. //点击box1之后做的事情 3. }); |

事件名一律不写on。特别的，鼠标进入改成了mouseenter，鼠标离开改为了mouseleave。

|  |
| --- |
| 1. oDiv.onclick = function(){ 3. } |

可以看做等价于：

|  |
| --- |
| 1. $("#box").click(function(){ 3. }) |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

# 目录

[jQuery框架](#_Toc22005)

[目录](#_Toc14956)

[一、复习上次课的内容](#_Toc27908)

[二、序与迭代](#_Toc22260)

[2.1 eq()方法](#_Toc28391)

[2.2 index()方法](#_Toc21751)

[2.3 each()](#_Toc19322)

[2.4 size()方法和length属性](#_Toc5770)

[2.5 get()方法](#_Toc28544)

[三、动画相关方法](#_Toc9308)

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

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

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

[3.4 stop()](#_Toc17400)

[3.5 finish()](#_Toc27716)

[3.6 delay()](#_Toc24907)

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

[四、节点关系](#_Toc28849)

[4.1 原生JS中nodeType属性](#_Toc5740)

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

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

[4.4 previousSibling、nextSibling](#_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。

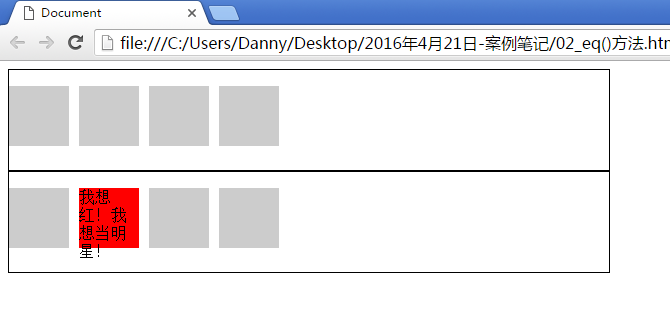
$()函数返回的是一组元素，每个方法比如css方法内部都有for循环迭代。说白了，不是没有for循环只不过for循环隐藏到css()方法里面了、隐藏到animate()方法里面了、隐藏到了click()方法里了。

# 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. }); |

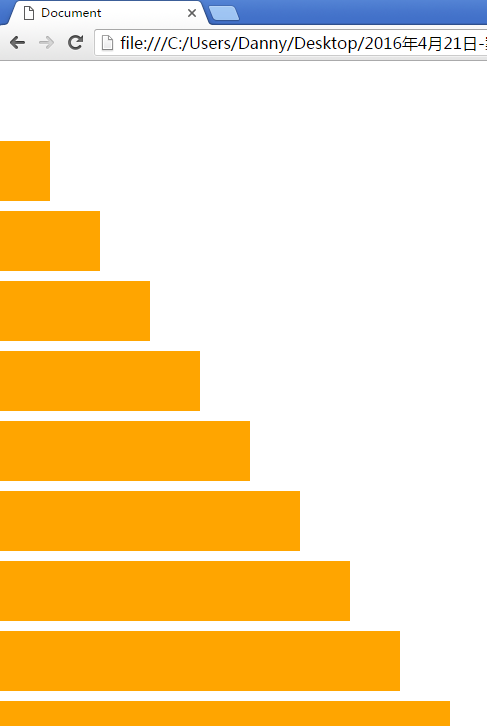
最后强调一下，是index()方法，而不是属性！

## 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. **}** |



# 目录

[jQuery框架](#_Toc21028)

[目录](#_Toc26977)

[一、认识框架](#_Toc10764)

[二、节点关系](#_Toc25810)

[2.1 原生JS中的节点关系](#_Toc20063)

[2.2 jQuery中的节点关系](#_Toc21391)

[三、节点操作](#_Toc4132)

[3.1 原生JavaScript节点操作](#_Toc228)

[3.1.1 createElement()和appendChild()](#_Toc23129)

[3.1.2 insertBefore](#_Toc11593)

[3.1.3 removeChild()](#_Toc12094)

[3.1.4 replaceChild()](#_Toc12034)

[3.1.5 cloneNode()](#_Toc32000)

[3.2 jQuery中的节点操作](#_Toc20630)

[3.2.1 append就是往最后添加：](#_Toc27859)

[3.2.2 appendTo 表示被动， 儿子appendTo父亲](#_Toc9231)

[3.2.3 after 插入兄弟 before 插入兄弟](#_Toc10165)

[3.2.4 insertBefore、insertAfter](#_Toc31010)

[3.2.5 wrap、warpAll](#_Toc9842)

[3.2.6 empty()](#_Toc21934)

[3.2.7 remove()](#_Toc3746)

[3.2.8 clone()](#_Toc2877)

# 一、认识框架

**计算机世界和物理化学不一样，计算机世界底层API少，逐渐一层一层的API丰富起来**。所谓的API就是你能够调用的方法、函数、语法。计算机的最底层是二进制，上面就是机器语言，再上面是汇编语言，再上面是高级语言。

汇编语言中，实际上就提供了不到100个API，比如汇编语言中求1~100的和：

|  |
| --- |
| 1. DSEG SEGMENT 2. RESULT DW ?;存放和 3. DSEG ENDS 4. CSEG SEGMENT 5. ASSUME CS:CSEG,DS:DSEG 6. START: 7. MOV AX,DSEG 8. MOV DS,AX 9. MOV AX,1;从1开始加 10. MOV RESULT,0;和的初值赋0 11. MOV CX,100;循环100次 12. NEXT: 13. ADD RESULT,AX;每次向和中累加AX 14. INC AX;然后AX加1 15. LOOP NEXT 16. MOV AX,4C00H;结束程序 17. INT 21H 18. CSEG ENDS 19. END START |

这里都是内存的变化，用内存的变化来编程。比如MOV表示把变量从内存一个地方移动到另一个地方。汇编语言非常晦涩，都是内存的变化。所以高级语言应运而生，C语言是最最著名的代表：

|  |
| --- |
| 1. int sum; 2. for(int i = 1 ; i <= 100 ; i++){ 3. sum += i; 4. } 5. prinf("%d",sum); |

c语言编译的过程，实际上就是把for这些字词翻译为汇编语言。大学里一门课程《编译原理》就是研究编译过程的。

上层的框架屏蔽了下层的语言的一些麻烦、不方便的东西，并且提供更方便的API。

jQuery就是干这个事情的，把JS中的不方便封装起来，暴露的API都是非常简便的。

**jQuery的哲学就是DOM编程领域的霸主，操作DOM节点、绑定监听、运动、css样式、Ajax等等都有封装。**

工作上都是用jQuery，如果不用jQuery也是用类似的东西。没有人会不用轮子去开发页面效果。

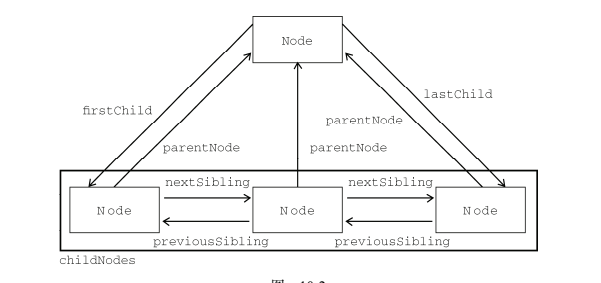
JavaScript中Library表示“库”，如果这个库的功能很强大，甚至颠覆了传统编程的语法、行文习惯，我们就可以叫做“框架”。

# 二、节点关系

## 2.1 原生JS中的节点关系

原生JS中提供的节点关系很少：

childNodes、firstChild、lastChild、parentNode、nextSibling、previousSibling



关于childNodes一定要记住，IE6、7、8和高级浏览器的不一致，高级浏览器视所有的换行为空文本节点，而IE6、7、8无视这个空文本节点。

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

|  |
| --- |
| 1. oDiv.childNodes.length; //chrome数值是9 IE6、7、8数值是4 |

为了解决兼容性问题（到底空文本算不算儿子，所以要用函数来解决）：

|  |
| --- |
| 1. //可以封装一个children函数，这个函数返回obj的所有节点儿子 2. function children(obj,num){ 3. var arr = []; //存放所有儿子 4. for(var i = 0 ; i < obj.childNodes.length ; i++){ 5. //寻找真正的HTML节点，不要考虑文本： 6. if(obj.childNodes[i].nodeType == 1){ 7. arr.push(obj.childNodes[i]); 8. } 9. } 10. //返回的是：如果用户传入了num，返回某一个儿子；如果没有num返回所有儿子 11. return num ? arr[num] : arr; 12. } |

同样的道理firstChild、lastChild也有兼容性问题。

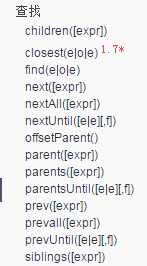
这个兼容性问题一直存在，比如前一个兄弟：

|  |
| --- |
| 1. <div id="box"> 2. <p></p> 3. <p></p> 4. <p></p> 5. <p></p> 6. <p id="xiaoming">小明</p> 7. <p></p> 8. <p></p> 9. <p></p> 10. </div> |

兼容问题：

|  |
| --- |
| 1. xiaoming.previousSibling; //chrome是空文本节点； IE6、7、8就会认为是上一个p |

## 2.2 jQuery中的节点关系



感恩的心，太好用了！！注意，都是方法！！因为里面暗含了迭代器了。

● children()方法

所有亲儿子节点。就是儿子，孙子不在children()里面。

让box的所有儿子变红：

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

儿子有很多，我们可以用选择器来表示怎么样的儿子：

|  |
| --- |
| 1. $("#box").children(**"h3"**).css("background-color","red"); |

还可以用筛选器：

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

jQuey屏蔽掉了浏览器兼容问题，不存在空文本节点也是儿子了，事实上children()返回的只能是节点，所有的文本都不算做是儿子了。

● find()

所有后代元素。和children()不一样，children()返回的是自己的亲儿子元素列表，而find()返回的是自己的后代所有元素的列表。

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

注意，和children()方法不一样，find()方法里面，必须写参数，表示后代的谁？

说白了find是寻找的意思，就是你要在后代寻找谁。

● parent()

亲爸爸。任何元素都只有一个亲爸爸。

|  |
| --- |
| 1. $("p").parent().css("background-color","red"); |

● parents()

这个元素的所有的祖先节点。

● siblings()

亲兄弟

|  |
| --- |
| 1. $(".xiaoming").siblings().css("background-color","red"); |

同样的，jQuery只返回节点元素，对于文本、注释都不视为兄弟。

可以加选择器、筛选器：

|  |
| --- |
| 1. $(".xiaoming").siblings(":odd").css("background-color","red"); |

排他，妈妈再也不用担心我写for循环了：

|  |
| --- |
| 1. $(this).addClass("cur").siblings().removeClass("cur"); |

我加cur，我的兄弟们去cur。

● prev()、next()、prevAll()、nextAll()

前一个兄弟、后一个兄弟、前所有兄弟、后所有兄弟。

总结：感恩！jQuery屏蔽了所有的不兼容，只选择nodeType=1的元素，不选择文本、注释等等。

并且不用for循环了，一次性得到所有东西了，原理我们后面的课程揭示。

# 三、节点操作

HTML节点我们原来最多最多就是改改HTML属性，比如src属性改改；或者改改css样式，比如.style或者.css()。

现在的问题是，我们要增加节点、删除节点、移动节点、替换节点。

## 3.1 原生JavaScript节点操作

### 3.1.1 createElement()和appendChild()

|  |
| --- |
| 1. var ul = document.getElementsByTagName("ul")[0]; 2. //创建一个li标签，用变量oLi来表示。创建出来的节点不是任何节点的儿子， 3. //也就是说没有在DOM树上， 4. var oLi = **document.createElement**("li"); 5. oLi.innerHTML = "DDDD"; //改变这个节点里面的内容 6. //把新创建的节点，追加到DOM树上 7. **ul.appendChild(oLi);** |

创建节点的API：

|  |
| --- |
| 1. document.createElement() |

create创建，Element元素。接收一个参数，就是创建的标签是什么。

追加节点的API：

创建出来的节点不在DOM树上，所以就应该用appendChild()来添加到DOM树上：

|  |
| --- |
| 1. 父亲.appendChild(新儿子); |

appendChild()一般来说就是用来追加新创建的节点。如果你试图把页面上已经有的节点，appendChild()到别的地方，那么这个节点将移动。也就是说，同一个节点不可能在页面上的两个地方出现。

比如结构：

|  |
| --- |
| 1. <div id="box1"> 2. <p id="xiaoming">我是小明</p> 3. </div> 4. <div id="box2"> 5. </div> |

语句：

|  |
| --- |
| 1. box2.appendChild(xiaoming); |

将使xiaoming移动位置，从box1里面移动到box2里面。

事实上，工作的时候很少使用createElement。因为innerHTML足够好用，innerHTML也可以用来创建节点，**甚至效率比createElement还高**。

|  |
| --- |
| 1. box.innerHTML = "<ul><li>哈哈哈</li><li>做顿饭</li></ul>"; |

### 3.1.2 insertBefore

我们刚才说的appendChild是把新节点在父亲的所有儿子后添加，也就是说添加的节点就是父亲的最后一个儿子。

我们可以在任意一个位置添加节点。

|  |
| --- |
| 1. 父亲.insertBefore(新儿子,原有标杆儿子); |

会在原有标杆儿子之前插入。

如果想每次添加都在开头添加，那么就是：

|  |
| --- |
| 1. ul.insertBefore(oLi, lis[0]); |

lis这个变量是动态的，这次添加的li，下回就是lis[0]

### 3.1.3 removeChild()

|  |
| --- |
| 1. 父亲.removeChild(儿子); |

如果要自杀，也要找到爸爸

|  |
| --- |
| 1. this.parentNode.removeChild(this); |

### 3.1.4 replaceChild()

替换节点

|  |
| --- |
| 1. 父亲.replaceChild(新儿子, 老儿子); |

### 3.1.5 cloneNode()

克隆节点，参数true表示深复制，节点里面的所有内容一同复制。

复制之后的节点是个孤儿节点，所以也需要使用appendChild、inserBefore、replaceChild来添加上DOM树。

|  |
| --- |
| 1. ul.appendChild(lis[0].cloneNode(true)); |

## 3.2 jQuery中的节点操作



### 3.2.1 append就是往最后添加：

父亲.append(儿子)

|  |
| --- |
| 1. $("#box").append("<p>么么哒</p>"); |

jQuery节点操作非常简单，不用createElement的过程了。实际上内部就是用innerHTML来给你加进去的。

特别的，如果我们想先创建一个孤儿节点，给这个孤儿节点加样式、加监听、加动画，可以的，那么创建孤儿节点，也是用$()函数。 $()不仅仅能够选择页面上的节点，也能够创建新的孤儿节点：

|  |
| --- |
| 1. var $obj = **$(**"<p></p>"**)**; 2. $obj.css("background-color","red"); //jQuery对象就可以直接调用css方法 3. $("#box").append($obj); |

### 3.2.2 appendTo 表示被动， 儿子appendTo父亲

|  |
| --- |
| 1. $("<p>么么哒</p>").appendTo($("#box")); |

prepend 插入为第一个儿子， prependTo表示被动

|  |
| --- |
| 1. $("<p>么么哒啊！</p>").prependTo($("#box")); |

### 3.2.3 after 插入兄弟 before 插入兄弟

比如HTML结构：

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

执行代码：

|  |
| --- |
| 1. $("p").after("<h3>我是h3</h3>"); |

HTML将变为：

|  |
| --- |
| 1. <div id="box"> 2. <p>哈哈</p> 3. <h3>我是h3</h3> 4. <p>嘻嘻</p> 5. <h3>我是h3</h3> 6. </div> |

### 3.2.4 insertBefore、insertAfter

比如HTML结构：

|  |
| --- |
| 1. <div id="box"> 2. <p>000</p> 3. <p>111</p> 4. <p>222</p> 5. <p>333</p> 6. <p>444</p> 7. <p>555</p> 8. </div> |

代码：

|  |
| --- |
| 1. $("<p>么么哒</p>").insertBefore($("p")[2]); |

HTML结构将变为：

|  |
| --- |
| 1. <div id="box"> 2. <p>000</p> 3. <p>111</p> 4. <p>么么哒</p> 5. <p>222</p> 6. <p>333</p> 7. <p>444</p> 8. <p>555</p> 9. </div> |

### 3.2.5 wrap、warpAll

在外面包裹

|  |
| --- |
| 1. $("a").wrap("<span></span>"); //每个a都用span包裹起来 2. $("a").wrapAll("<span></span>"); //所有a都用span包裹起来 |

### 3.2.6 empty()

|  |
| --- |
| 1. $("p").empty(); |

等价于

|  |
| --- |
| 1. $("p").html(“”); |

### 3.2.7 remove()

删除页面上所有p标签：

|  |
| --- |
| 1. $("p").remove(); |

### 3.2.8 clone()

克隆节点

克隆ul li里面的第1个，然后ul追加：

|  |
| --- |
| 1. $(“ul”).append($(“ul li”).eq(0).clone()); |

反过来：

|  |
| --- |
| 1. $(“ul li”).eq(0).clone().appendTo($(“ul”)); |

不要背API，常用的慢慢就记住了。要思考业务逻辑。