爱创课堂前端培训

jquery第二天

班级：爱创课堂十七期

讲师：彭帅伟

日期：2018年11月1日

目录

[复习： 2](#_Toc16430)

[一、 序号的问题 4](#_Toc18166)

[1.1 eq() 4](#_Toc407)

[1.2 index 6](#_Toc22327)

[1.3 对应思想 7](#_Toc11903)

[1.4 each() 7](#_Toc9756)

[下午复习： 9](#_Toc29973)

[二、 animate函数 9](#_Toc7214)

[2.1 概述 9](#_Toc2694)

[2.2 能够参与动画的属性 10](#_Toc30084)

[2.3 动画的执行顺序 11](#_Toc12201)

[2.4 回调函数 12](#_Toc30796)

[2.5 delay() 13](#_Toc16349)

[2.6 stop() 14](#_Toc12461)

[三、 防止动画积累 15](#_Toc17385)

# 复习：

jquery引包：引入jquery的script标签不能与书写jquery语句的script是同一个

$函数： 引入jquery之后， 向全局暴露一个$函数， 所以我们书写的所有jquery语句都是以$开头

css选择器：

支持Css2.1选择器

$(“.box”)

$(“#box”)

$(“div”)

$(“\*”)

$(“div h1”)  
 $(“div.box”)

$(“div, h3”)

Css3选择器：

$(“p:first”)

$(“p:last”)

$(“p:eq(x)”)

$(“p”).eq(x)

$(“p:odd”)

$(“p:even”)

$(“p:lt(x)”)

$(“p:gt(x)”)

常用方法：

size: 表示获取元素对象的个数

常用的是length属性

css: 读取元素计算后的样式，参数就是要读取的属性名， 返回值就是属性值

css: 可以设置元素的样式属性，

设置单一属性

对象.css(key, value)

设置多个属性

对象.css({

k: v,

k: v

})

html: 可以获取元素的内部文本，不需要书写参数

可以设置元素的内部文本， 接收一个参数， 就是要设置的内容

添加事件的方式：

第一种方式：

$(dom).on(type, fn)

type: 事件类型

fn: 事件函数

第二种方式:

$(dom).click(function() {})

show(): 显示的意思， 只有当元素的状态是display: none的时候才可以调用

hide(): 隐藏的意思， 与show()相反

toggle: 切换

slideDown(): 慢慢展开，只有当元素的状态是display: none的时候才可以调用

slideUp(): 徐徐合起， 与slideDown相反

slideToggle: 切换

fadeIn(): 淡入，只有当元素的状态是display: none的时候才可以调用

fadeOut(): 淡出， 与fadeIn()相反

fadeToggle: 切换

fadeTo(600, .5): 第一个参数表示动画时间， 第二个参数是透明度

addClass(): 添加类名

removeClass(): 移除类名

attr(): 可以设置元素的原有属性， 接受两个参数， 属性名， 属性值

如果传递一个参数，表示读取

节点操作：

this: 触发事件的对象

$(this)

$(window)

$(document)

parent(): 父节点， 只会选中亲父亲， 不会选中爷爷或者是曾爷爷

siblings(): 兄弟节点

children(): 子节点

next(): 下一个兄弟节点

nextAll(): 后面所有兄弟

prev(): 上一个兄弟节点

prevAll(): 前面所有的兄弟

parents(): 祖先节点

find(): 后代节点

# 序号的问题

## 1.1 eq()

|  |
| --- |
| 1. <div class="box" id="box"> 2. <p></p> 3. <p></p> 4. <p></p> 5. <p></p> 6. </div> 7. <div class="box"> 8. <p></p> 9. <p></p> 10. <p></p> 11. <p></p> 12. </div> 13. <div class="box"> 14. <p></p> 15. <p></p> 16. <p></p> 17. <p></p> 18. </div> 19. <div class="box"> 20. <p></p> 21. <p></p> 22. <p></p> 23. <p></p> 24. </div> 25. <script type="text/javascript" src="js/jquery-1.12.3.min.js"></script> 26. <script type="text/javascript"> 27. // 点击哪个div就让哪个div的第一个儿子背景色改变 28. $("div").click(function() { 29. $(this).children().eq(0).css("backgroundColor", "red"); 30. }) 31. </script> |

执行代码：

|  |
| --- |
| 1. // 点击哪个div就让哪个div的第一个儿子背景色改变 2. $("div").click(function() { 3. $(this).children().eq(0).css("backgroundColor", "red"); 4. }) |

结果：

|  |
| --- |
|  |

eq(): 选中元素大队列的排名，与亲兄弟没有直接关系

|  |
| --- |
| 1. $("div").click(function() { 2. $("p").eq(0).css("backgroundColor", "red"); 3. }) |

eq(): 选中元素的不同， 序号也是不同的

结构：

|  |
| --- |
| 1. <div class="box" id="box"> 2. <p></p> 3. <p class="teshu"></p> 4. <p></p> 5. <p class="teshu"></p> 6. </div> 7. <div class="box" id="box1"> 8. <p></p> 9. <p></p> 10. <p class="teshu"></p> 11. <p class="teshu">我想变红</p> 12. </div> |

执行代码：

|  |
| --- |
| 1. // $("p").eq(7).css("background-color", "red"); 2. // $("#box1 p").eq(3).css("background-color", "blue"); 3. $("p.teshu").eq(3).css("background-color", "orange"); |

## 1.2 index

index: 表示亲兄弟的排名，无视亲兄弟的类型， 只要是同一个父节点就是亲兄弟

DOM结构：

|  |
| --- |
| 1. <div class="box" id="box"> 2. <ul> 3. <li>li</li> 4. </ul> 5. <h3>h3</h3> 6. <p>1</p> 7. <p>2</p> 8. <p>3</p> 9. <p id="p1">弹出亲兄弟的排名</p> 10. </div> |

执行代码：

|  |
| --- |
| 1. // 点击p1弹出亲兄弟的排队 2. $("#p1").click(function() { 3. console.log($(this).index()); 4. }) |

结果：

|  |
| --- |
|  |

index(): 它的值非常稳定， 无视选择器的选择， 都是弹出3

结构：

|  |
| --- |
| 1. <div class="box" id="box"> 2. <p>1</p> 3. <p class="teshu">2</p> 4. <p class="teshu">3</p> 5. <p>4</p> 6. </div> 7. <div class="box" id="box1"> 8. <p>1</p> 9. <p class="teshu">2</p> 10. <p>3</p> 11. <p class="teshu">弹出亲兄弟的排名</p> 12. </div> |

代码：

|  |
| --- |
| 1. $("p").click(function() { 2. console.log($(this).index()); 3. }) 4. $(".teshu").click(function() { 5. console.log($(this).index()); 6. }) 7. $("#box1 p").click(function() { 8. console.log($(this).index()); 9. }) |

结果都是弹出3：

|  |
| --- |
|  |

## 1.3 对应思想

编程中最重要的一点就是对应思想

结构：

|  |
| --- |
| 1. <div class="box" id="box"> 2. <p></p> 3. <p></p> 4. <p></p> 5. <p></p> 6. </div> 7. <div class="box" id="box1"> 8. <p></p> 9. <p></p> 10. <p></p> 11. <p></p> 12. </div> |

执行代码：

|  |
| --- |
| 1. // 点击box p 让 box1 p 对应的元素背景色改变 2. $("#box p").click(function() { 3. // console.log($(this).index()); 4. $("#box1 p").eq($(this).index()).css("backgroundColor", "red"); 5. }) |

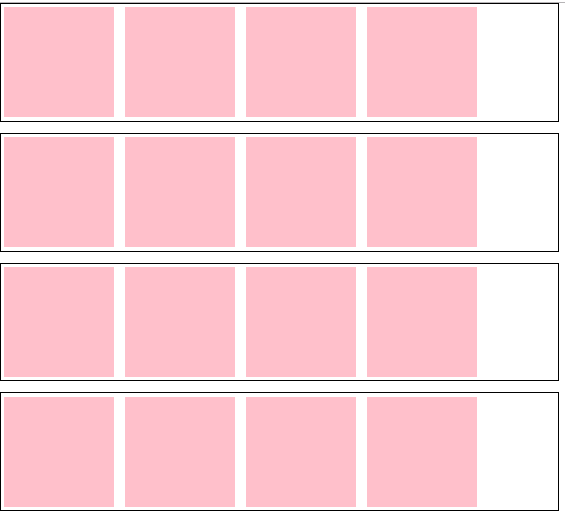
## 1.4 each()

each(): 表示“每一个”， 会循环、遍历所匹配到的所有元素， 接受一个参数是匿名函数， 函数中也有this,

表示遍历到的该对象

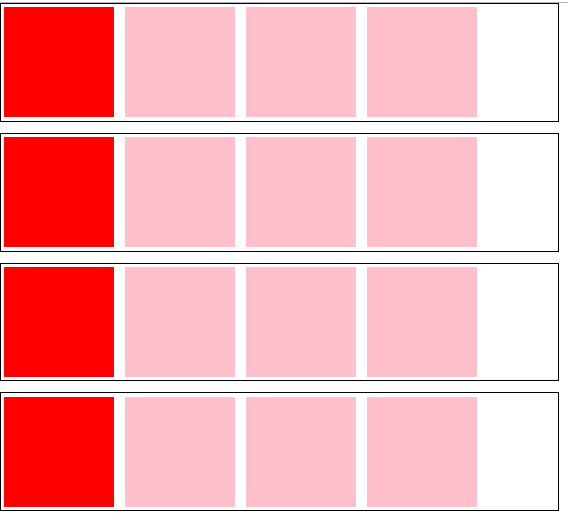
结构：

|  |
| --- |
| 1. <div class="box"> 2. <p></p> 3. <p></p> 4. <p></p> 5. <p></p> 6. </div> 7. <div class="box"> 8. <p></p> 9. <p></p> 10. <p></p> 11. <p></p> 12. </div> 13. <div class="box"> 14. <p></p> 15. <p></p> 16. <p></p> 17. <p></p> 18. </div> 19. <div class="box"> 20. <p></p> 21. <p></p> 22. <p></p> 23. <p></p> 24. </div> |



执行代码：

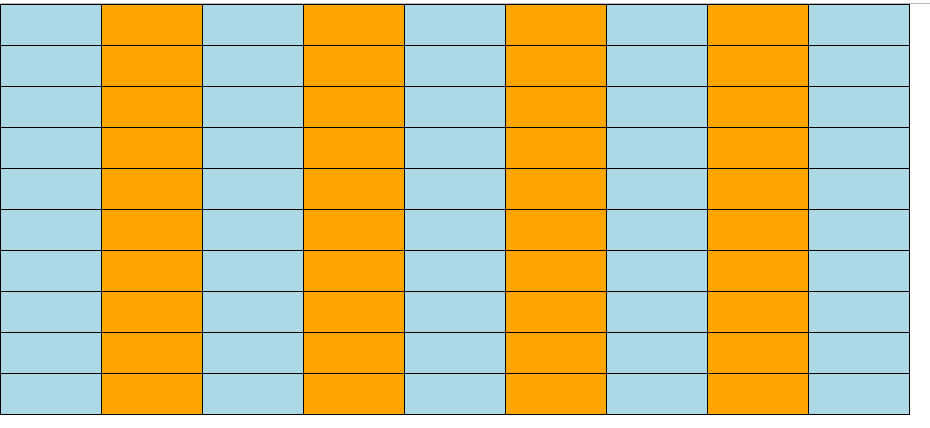
|  |
| --- |
| 1. // 让每一个div中的第一个儿子背景色改变 2. $(".box").each(function() { 3. // console.log($(this)); 4. // console.log(this); 5. $(this).children().eq(0).css("background-color", "red"); 6. }) |



当看到“每一个”要做什么事情， 首先要想到each方法

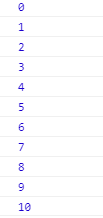
执行代码：

|  |
| --- |
| 1. // 实现隔列变色 2. $("tr").each(function() { 3. // console.log(this); 4. $(this).children("td:odd").css("backgroundColor", "orange"); 5. $(this).children("td:even").css("backgroundColor", "lightblue"); 6. }) |



each方法中的参数是一个函数，函数中有一个参数i, 表示遍历到该对象的序号

|  |
| --- |
| 1. // 输出i 2. $(".teshu").each(function(i) { 3. // console.log(i); 4. }) 5. $("tr").each(function(i) { 6. console.log(i); 7. }) |



# 下午复习：

序号的问题：

eq(): 表示选中元素大队列的排名， 选中元素不同，序号也是不相同的

index(): 表示亲兄弟的排名， 它的值非常稳定，无视亲兄弟的类型

each(): 表示“每一个”， 会遍历，循环所有匹配到的元素， 接受一个 参数 是匿名函数， 函数中的this，表示遍历到的该对象， 函数中有一个参数i， 表示遍历到的该对象的序号

# animate函数

## 2.1 概述

animtae: 可以接受两个 参数

第一个参数： json

第二个参数： 表示完成动画的时间， 单位是ms, 省略不写

animate是基于终点的动画

结构：

|  |
| --- |
| 1. <p></p> 2. <p></p> 3. <p></p> 4. <p></p> |

执行代码：

|  |
| --- |
| 1. // 使每一个P元素动画到500的位置 2. $("p").animate({"left": 500}, 2000); |

结果：

|  |
| --- |
|  |

## 2.2 能够参与动画的属性

所有数值型的属性都能够参与动画

比如：width

height

border

不能够参与动画的属性

background-color

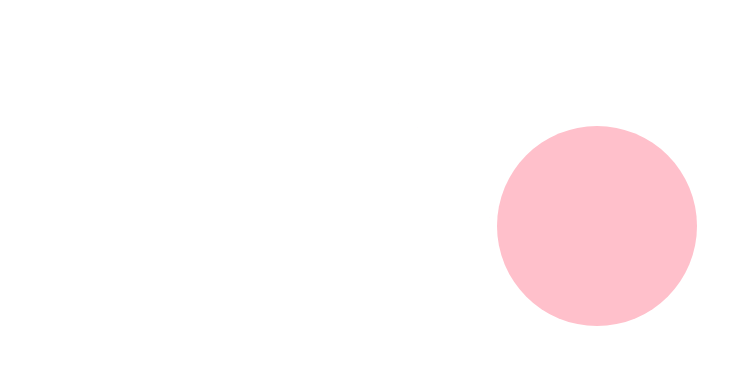
background-position

Css3的一些复杂属性（transform）

border-radius 是可以参与动画

举例：

|  |
| --- |
| 1. $("p").animate({ 2. "width": 200, 3. "height": 200, 4. "left": 500, 5. "top": 500, 6. "backgroundColor": "orange", 7. "border-radius": "50%" 8. }, 2000); |



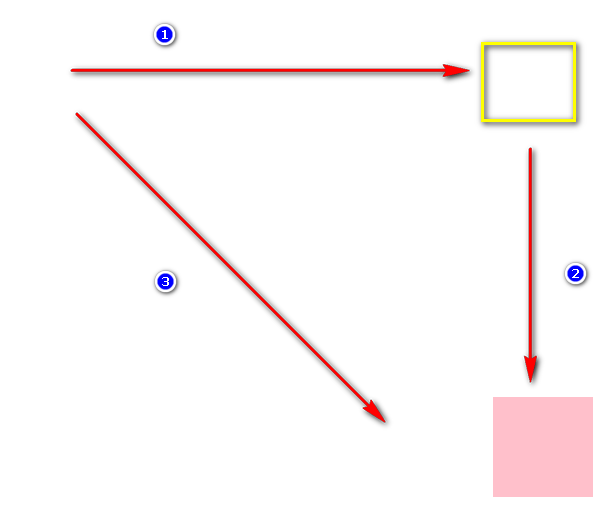
## 2.3 动画的执行顺序

* 同一元素的不同animate, 动画执行顺序是按照绑定的顺序执行

|  |
| --- |
| 1. // 绑定第一个动画 2. $("p").animate({"left": 500}, 2000); 3. // 绑定第二个动画 4. $("p").animate({"top": 500}, 2000); |

不等价于：

|  |
| --- |
| 1. // 不等价于 2. $("p").animate({ 3. "left": 500, 4. "top": 500 5. }, 2000) ③ |



* 不同元素的animate执行顺序，是同时执行

|  |
| --- |
| 1. // 绑定动画事件 2. $("p").animate({"left": 500}, 2000); 3. $(".box").animate({"left": 500}, 2000); |



* animate和css

animate()：需要花费时间

css(): 瞬间完成

|  |
| --- |
| 1. // 让p元素运动完毕之后改变颜色 2. $("p").animate({"left": 500}, 2000).css("backgroundColor", "red"); |

结果：

|  |
| --- |
|  |

## 2.4 回调函数

异步语句： 类似于animate这种需要花费时间的语句，如果后面还有其它js语句， 这些js语句会立即执行，不会死等着animate执行完毕之后再去执行。

异步语句一般都有一个回调函数，异步语句执行完毕之后要做的事就是写在回调函数中

比如：我们要做一个肉菜， 需要解冻肉

把肉放在微波炉中进行解冻，需要花费10min， 在这期间我们还可以洗菜， 摘菜， 烧水， 当微波炉叮一声之后（就是回调函数）， 将肉拿出，开始做肉菜， 做肉菜的行为就是书写在回调函数。

animate(), show(1000), hide(1000), slideDown(), slideUp(), fadeIn(), fadeOut()这些都是异步语句

实际上animate可以接受三个参数：

第一个参数： json

第二个参数：完成动画的时间， 单位ms ， 省略不写

第三个参数： 回调函数

举例：

|  |
| --- |
| 1. // 让p元素运动完毕之后改变背景色 2. $("p").animate({"left": 500}, 2000, function() { 3. $(this).css("background-color", "red"); 4. }) |

结果：

|  |
| --- |
|  |

比如： fadeIn也可以书写回调函数

|  |
| --- |
| 1. $(".box").fadeIn(2000, function() { 2. $(this).css("background-color", "blue"); 3. }) |

## 2.5 delay()

delay: 表示延迟， 只能在动画的前面来使用, 接受一个参数就是要延迟的时间， 单位是ms, 省略不写

animate(), show(1000), hide(1000), slideDown(), slideUp(), fadeIn(), fadeOut()

举例：

|  |
| --- |
| 1. // 点击btn 让box延迟2s之后再去运动 2. $("#btn").click(function() { 3. $(".box").delay(2000).animate({"left": 1000}, 2000, function() { 4. $(this).css("backgroundColor", "red"); 5. }) 6. }) |

比如: show也可以书写delay

|  |
| --- |
| 1. // 点击btn让box1延迟2s再出现 2. $("#btn").click(function() { 3. $(".box1").delay(2000).show(2000, function() { 4. $(this).css("background-color", "blue"); 5. }) 6. }) |

利用delay控制元素的进场顺序

结构：

|  |
| --- |
| 1. <p></p> 2. <p></p> 3. <p></p> 4. <p></p> 5. <p></p> 6. <p></p> 7. <p></p> 8. <p></p> |

执行代码：

|  |
| --- |
| 1. // 利用delay控制元素进场顺序 2. $("p").each(function(i) { 3. // console.log(this); 4. // console.log(i); 5. $(this).delay(500 \* i).show(0); 6. }) |

结果：

|  |
| --- |
|  |

## 2.6 stop()

stop(): 表示让运动的元素立即停止当前动画

接受两个参数， 都是布尔值，默认是false

第一个参数： 是否清空当前动画队列

第二个参数： 是否立即完成当前动画

|  |
| --- |
| 1. stop(false, false) 等价方式stop(), 表示立即进入下一个动画， 表示立即停止当前动画 2. stop(false, true) 表示立即进入下一个动画， 立即停止并完成当前动画 3. stop(true, true) 表示清空当前动画队列， 立即停止并完成当前动画 4. stop(true, false) 等价方式stop(true) 表示清空当前动画队列， 立即停止当前动画 |

|  |
| --- |
| 1. // 添加事件 2. // stop(false, false) 3. $("#btn").click(function() { 4. $(".box").stop(); 5. }) 6. // stop(false, true) 7. $("#btn1").click(function() { 8. $(".box").stop(false, true); 9. }) 10. // stop(true, true) 11. $("#btn2").click(function() { 12. $(".box").stop(true, true); 13. }) 14. // stop(true, false) 15. $("#btn3").click(function() { 16. $(".box").stop(true); 17. }) |

# 防止动画积累

一个元素身上可能存在多个动画事件，尤其在不经意间造成的动画积累，这个时候我们就要进行防止（也叫作防流氓）

比如：现在有一支队伍在北京，前往广州， 在前往广州的路上又接到新的命令， 回北京

此时， 这支队伍有两个策略：

①：立即放弃原有任务， 执行新的任务

利用stop(true)来实现

|  |
| --- |
| 1. // 当点击btn1的时候 运动到1000的位置 2. $("#btn1").click(function() { 3. $(".box").stop(true).animate({"left": 1000}, 2000); 4. }) 5. // 当点击btn的时候 恢复到原来的位置 6. $("#btn").click(function() { 7. $(".box").stop(true).animate({"left": 50}, 2000); 8. }) |

②： 只有当元素不运动的时候，才可以接受新的任务

is(): 表示元素是否具备某种状态

|  |
| --- |
| 1. is(“:animated”): 当元素在运动的时候会返回true, 当元素不运动的时候返回false |

|  |
| --- |
| 1. // 当元素运动的时候， 不接受新的任务 2. $("#btn1").click(function() { 3. if ($(".box").is(":animated")) { 4. return; 5. } 6. // 元素不运动的时候， 才接受新任务 7. $(".box").animate({"left": 1000}, 2000); 8. }) 9. // 当元素不运动的时候， 才接受新的任务 10. $("#btn").click(function() { 11. if (!$(".box").is(":animated")) { 12. // 让元素运动 13. $(".box").animate({"left": 50}, 2000); 14. } 15. }) |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |