准备: 微信小程序: WEB问题速查 小恐龙图标

1. 历届师兄师姐在今天课程中曾经问过的高频的经典的问题清单和解答: 小程序->首页->JQUERY-> day01、day02

自己可能没有发现问题，但是想知道别的同学都问过哪些问题

2. 今日课程对应的重点案例的视频以及鄙视面试所需的扩展知识视频：小程序->在线->JQUERY-> day01、day02

(学有余力!!!的同学或自己认为自己基础好的同学必须看!!!!!!)：

今日扩展视频：

小程序->在线->JQUERY->day02

作业: jQuery单击下拉菜单

作业: jQuery带多种动画效果的对话框

直接在day02/5\_class2.html中做即可！

12. jQuery事件 事件委托

3. 今日课程中包含的英语单词的翻译：小程序->单词->JQUERY->day01、02

英文不太好的同学，可以提前看一下第二天需要用到哪些英文单词。

约定: 2件事:

1. 第三阶段禁止上课跟着敲笔记和敲代码！谁抄笔记抄的勤，谁抄代码抄的勤，谁学的越烂！

2. 每个完整功能的案例/视频，至少写三遍: (如果需要画图，必须亲手画图)

(1). 连代码+注释抄一遍 —— 明白这个例子是要做什么事儿

(2). 保留注释，自己试着看着注释，将代码填回来 —— 复习函数和对象

(3). 删除所有代码和注释，自己试着用自己的话，把注释写回来，再翻译为代码！—— 只有自己可以写出人话的注释，这个例子才算自己会！

如果没有问题的同学: （享受vip一对一答疑的时间还剩8天）

1. 上课时那句话没听懂，笔记中那句话看不懂，案例中那句话看不懂，立刻问！

2. 小程序问题清单中哪个问题或解答看不懂，立刻问！

3. 小程序视频中哪个视频看不懂，立刻问

4. 个人项目中哪个功能不会做，立刻问

5. 平时刷笔试题时，遇到不会的笔试题，立刻问

谁问的问题越多！谁的培训费交的越值！

正课:

1. 查找

2. 修改

3. 按节点间关系查找

4. 添加/删除/替换/克隆

5. 事件绑定

一. 查找: jQuery新增选择器

对应小程序视频: 小程序->在线->JQUERY->day01 4. jQuery其它选择器...

1. 内容过滤选择器:

(1). 什么是: 根据元素的内容中包含的关键词来匹配元素

(2). 何时: 如果用其它选择器很难区分元素时，就可以尝试用元素的内容不同来选择元素

(3). 包含: 4种:

a. :contains(关键词) 匹配元素内容中包含指定"关键词"的元素

b. :has(选择器) 匹配子元素中包含符合选择器要求的元素 的父元素

根据孩子特征，选爹

c. :parent 匹配所有内容不为空的元素

d. :empty 匹配内容为空的元素

强调: 空元素: 连空字符内容都不能有！

(4). 示例: 使用内容过滤选择器选择指定元素

day01 剩余/10\_content filter.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">  <meta charset="UTF-8">  <title></title>  <style>  div {  margin: 10px auto;  }  </style>  </head>  <body>  <div class="container">  <h1>jQuery中的选择器——内容过滤选择器</h1>  <button>提交订单</button>  <button>Submit注册信息</button>  <button>马上提交</button>  <button>清空重填</button>  <hr />  <div class="alert" id="alert1">  第一个警告框  </div>  <div class="alert" id="alert2">  <span class="close">×</span>  第一个警告框  </div>  <div class="alert" style="height:30px"></div>  </div>  <script src="js/jquery-1.11.3.js"></script>  <script>  //选择包含"提交"二字的按钮，变为绿色按钮  $(".container>:contains(提交)")  .css("background-color", "lightGreen");  //选中包含.close按钮的.alert元素，让它们变为红色的警告框  $(".alert:has(.close)").css("background-color", "pink")  //选中不包含.close按钮的.alert元素，让它们变为绿色的警告框  $(".alert:not(:has(.close))")  .css("background-color", "lightGreen");  //希望给内容为空的div加红色边框  $(".alert:empty").css("border", "1px solid red");  //希望给有内容的div加绿色边框  $(".alert:parent").css("border", "1px solid green");  </script>  </body>  </html>  运行结果: |

2. 可见性过滤:

(1). 什么是: 根据元素是否可见来选择元素

(2). 包括:

a. :visible 专门匹配可见的元素

b. :hidden 专门匹配不可见的元素

坑: 只能选择display:none和type="hidden"隐藏的元素

~~无法选择visibility:hidden和opacity:0的元素~~

(4). 示例: 使用:hidden查找隐藏的元素

day01 剩余/12\_visibility filter.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">  <meta charset="UTF-8">  <title></title>  <style>  </style>  </head>  <body>  <div class="container">  <h1>可见性选择器</h1>  <div id="d1" style="display: none">lorem</div>  <div id="d2" style="visibility: hidden">lorem</div>  <div id="d3" style="opacity: 0">lorem</div>  <input id="d4" type="hidden" name="aa" value="bb" />  </div>  <script src="js/jquery-1.11.3.js"></script>  <script>  //想选择class为container下的隐藏的元素  var $elems=$(".container>:hidden");  console.log($elems);//4个元素  </script>  </body>  </html>  运行结果: |

3. 表单元素过滤选择器:

(1). :input 专门选择所有表单元素(input 、button、textarea 、select)

vs input 只是一个普通的元素选择器，只能选择input元素。不能选择其他类型的表单元素

(2). 问题: input元素就有很多种类，如何精确的只选择其中一种种类的input呢？

(3). 解决: 其实，input元素的每种type值，都对应着一个专门的选择器，比如:

a. :text 专门选择<input type="text"

b. :password 专门选择<input type="password"

c. :radio 专门选择<input type="radio"

... ...

(4). 示例: 点同意，启用表单元素；不同意，就禁用表单元素

day01 剩余/13\_form state selector.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>.....</title>  <meta charset="utf-8" />  <style>  </style>  </head>  <body>  <form>  用户名:<input disabled></br>  密码:<input type="password" disabled></br>  <input type="checkbox">我同意本站的使用条款<br>  <input type="submit" value="提交注册信息" disabled />  </form>  <script src="js/jquery-1.11.3.js"></script>  <script>  //DOM 4步  //1. 查找触发事件的元素  //本例中: 用户点type=checkbox的复选框触发启用禁用状态的改变  //应该查找type=checkbox的一个复选框  $(":checkbox")  //2. 绑定事件处理函数  //本例中:单击复选框触发变化  .click(function(){  //3. 查找要修改的元素  //本例中: 点了复选框要修改除当前复选框之外其余所有表单元素  var $others=$(":input:not(:checkbox)")  //4. 修改元素  //获得当前checkbox的选中状态  //错误: jQuery是函数库，不能用属性方式  //var checked=$(this).checked;  // 当前复选框 获取 checked属性的值  var checked=$(this).prop("checked");  //因为checked属性值只是一个bool值，不是jQuery家子对象  //所以变量名不加$  //如果checkbox是选中的  if(checked==true){  //其它表单元素启用  //修改其他表单元素的disabled属性值为false  $others.prop("disabled",false);  }else{//否则如果checkbox是未选中的  //其它表单元素禁用  //修改其他表单元素的disabled属性值为true  $others.prop("disabled",true);  }  //基础好的同学，必须改为1句话！  //如果选中(checked为true)，则其它元素启用(disabled为false)  //否则如果未选中(checked为false)，则其它元素禁用(disabled为true)  //所以，只要让其他元素的disabled值和checked的值相反就行！不用三目！  })  </script>  </body>  </html>  运行结果: |

二. 修改: 3种东西

1. 内容: 3种:

对应小程序视频: 小程序->在线->JQUERY->day02 1. jquery 修改元素的内容(有扩展)

(1). 原始的HTML内容:

a. DOM中: 元素.innerHTML

b. jq中: $元素.html("新内容")

(2). 纯文本内容:

a. DOM中: 元素.textContent

b. jq中: $元素.text("新内容")

(3). 表单元素的值:

a. DOM中: 元素.value

b. jq中: $元素.val("新值")

(4). 示例: 通过修改元素内容，实现表单验证

1\_html\_val.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title> new document </title>  <meta charset="utf-8">  </head>  <body>  <h1>操作元素的内容和值</h1>  <form action="">  用户名:<input name="uname">  <span></span><br>  密码:<input type="password" name="upwd">  <span></span><br>  <input type="submit" value="提交注册信息">  </form>  <script src="js/jquery-1.11.3.js"></script>  <script>  //正确时，使用图片:"<img src='img/ok.png'>"  //姓名错误时: "<img src='img/err.png'>用户名必须介于3~9位之间!"  //密码错误时: "<img src='img/err.png'>密码必须介于6~8位之间!"  //DOM 4步  //1. 查找触发事件的元素  //本例中: 当姓名文本框失去焦点时，触发验证和变化  //应该找文本框:  $(":text")  //2. 绑定事件处理函数  //本例中: 失去焦点触发验证和变化  .blur(function(){  var $txt=$(this);  //3. 查找要修改的元素  //本例中: 不管验证通过还是未通过，都要修改文本框旁边的下一个span  var $span=$txt.next();  //4. 修改元素  //获得文本框中内容去掉空字符后的长度  var length=$txt.val().trim().length;  //如果长度介于3~9位字符之间，就验证通过  if(length>=3&&length<=9){  //就设置span的内容为<img src='img/ok.png'>  $span.html(`<img src='img/ok.png'>`)  }else{//否则如果验证不通过  //就设置span的内容为<img src='img/err.png'>用户名必须介于3~9位之间!  $span.html(`<img src='img/err.png'>用户名必须介于3~9位之间!`);  }  });  //自己模仿我的姓名文本框，制作密码框验证  </script>  </body>  </html>  运行结果: |

2. 属性: 3种:

对应小程序视频: 小程序->在线->JQUERY->day02 2. jquery修改元素属性...

(1). 字符串类型的HTML标准属性:

a. DOM中: 2种:

1). 核心DOM 4个函数: 元素.getAttribute() setAttribute() ...

2). HTML DOM: 元素.属性名

b. jq中: 2种:

1). $元素.attr("属性名", "新值") 代替元素.getAttribute() setAttribute() ...

attribute指 HTML中开始标签中的属性

一个函数两用

2).问题: jQuery是函数库，不能用".属性名"方式访问

解决: jQuery中封装了一个新函数.prop("属性名", "新值")，专门代替".属性名"方式来获取或修改元素的属性值

property 指 内存中元素对象上的属性

一个函数两用

(2). bool类型的HTML标准属性

a. DOM中: 元素.属性名， ~~不能用核心DOM getAttribute()和setAttribute()~~

d. 如何: $元素.prop("属性名", bool) 代替".属性名"

~~c. 因为在DOM中不能用getAttribute()和setAttribute()，所以在jq中也不能用attr()~~

(3). 自定义扩展属性:

a. DOM中: 2种:

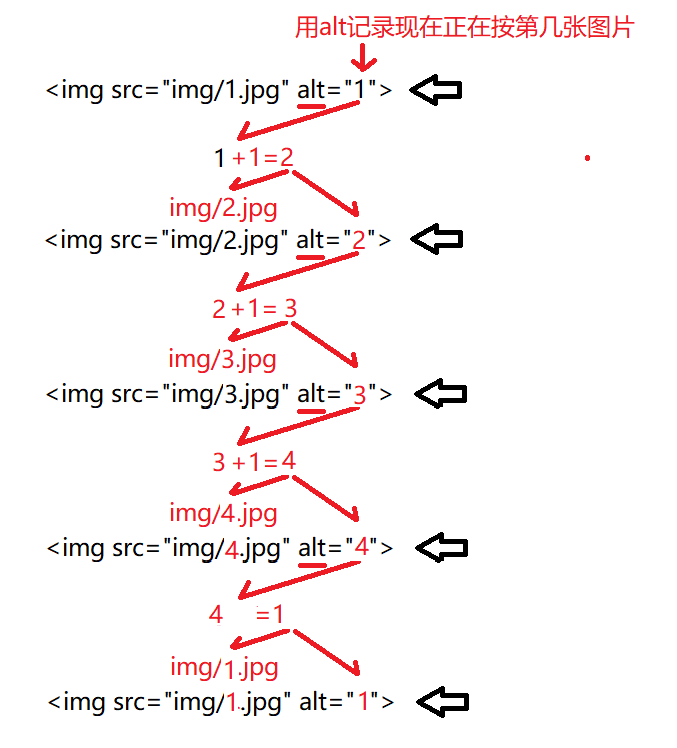
1). 核心DOM4个函数: 元素.getAttribute() 元素.setAttribute()

2). HTML5: 元素.dataset.自定义属性名

b. jq中: 1种: $元素.attr() 代替 元素.getAttribute() 元素.setAttribute()

~~c. 强调: 因为自定义属性在DOM中就不能用".属性名"方式访问，所以jq中自然也就不能用.prop()来访问！因为prop()代替的就是".属性名"~~

(4). 示例: 点击图片切换下一张



2\_attr.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title> new document </title>  <meta charset="utf-8">  </head>  <body>  <h1>操作元素的属性</h1>  <img src="img/1.jpg" alt="1">  <script src="js/jquery-1.11.3.js"></script>  <script>  //单击图片，切换下一张  //DOM 4步  //1. 查找触发事件的元素  //本例中: 用户点img触发变化  $("img")  //2. 绑定事件处理函数  //本例中: 单击触发变化  .click(function(){  //3. 查找要修改的元素  //本例中: 就是要修改当前img自己  var $img=$(this);  //4. 修改元素  //本例中:  //先获取当前img中alt属性的值转为整数  var i=parseInt($img.attr("alt"));  // .prop("alt")  if(i<4){//如果alt值<4，则alt+1  i++;  }else{//否则如果alt值==4，则alt=1  i=1;  }  //最后将alt的值拼成新的src路径，设置到img的src属性上，同时，将alt值再放回img的alt属性中，为下次点击做准备  //$img.attr("src",`img/${i}.jpg`);  // .prop()  //$img.attr("alt",i);  // .prop()  $img.attr({ src:`img/${i}.jpg`, alt:i })  // .prop()  })  </script>  </body>  </html>  运行结果: |

(5). 示例: 点击小图片，切换大图片

3\_attr2.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">  <meta charset="UTF-8">  <title></title>  <style>  body {  text-align: center;  }  </style>  </head>  <body>  <img src="img/1.jpg" data-target="img/1-l.jpg" class="my-small">  <img src="img/2.jpg" data-target="img/2-l.jpg" class="my-small">  <img src="img/3.jpg" data-target="img/3-l.jpg" class="my-small">  <img src="img/4.jpg" data-target="img/4-l.jpg" class="my-small">  <hr />  <img src="img/1-l.jpg" class="my-big">  <script src="js/jquery-1.11.3.js"></script>  <script>  //点击小图片，下方my-big中显示大图片  //DOM 4步  //1. 查找触发事件的元素  //本例中: 用户点class为my-small的img元素触发变化  $(".my-small")  //2. 绑定事件处理函数  //本例中: 单击触发变化  .click(function(){  //3. 查找要修改的元素  //本例中: 要修改class为my-big的img  $(".my-big")  //4. 修改元素  //本例中: 要修改my-big的src属性为当前点击的小图片上的data-target属性值！  // 当前小图片 获取 data-target属性值  .attr("src",$(this).attr("data-target"))  //.prop() 不能改.prop()  })  </script>  </body>  </html>  运行结果： |

3. 样式:

对应小程序视频: 小程序->在线->JQUERY->day02 3. jQuery修改样式 css()...

(1). DOM中:

a. 如果只修改一个css属性: 元素.style.css属性=值

b. 获取一个元素的css属性: var style=getComputedStyle(元素)

c. 如果批量修改一个元素的多个css属性: 元素.className="class名"

(2). jq中:

a. 将修改css属性和获取css属性统一为一个: $元素.css("css属性名","属性值")

一个函数两用:

如果没传入新属性值，则自动执行读取旧属性值的操作，相当于getComputedStyle

如果传入新属性值，则自动切换为执行修改属性值的操作，相当于元素.style.css属性

b. 如果批量修改一个元素的多个css属性，依然使用class的方式。

1). 问题: DOM中的className如果只想操作一个class名而不影响当前元素上其它class名，非常不方便！因为className是一个完整的字符串，只能整体替换所有class。

2). 解决: jq定义两个4个函数，专门对class执行不同的操作：

i. $元素.addClass("class名")

添加class

ii. $元素.removeClass("class名")

移除class

iii. var bool=$元素.hasClass("class名")

判断是否包含某个class

iv. $元素.toggleClass("class名")

切换有没有一个class

①等效于:

if(有这个class hasClass()){

就移除class removeClass()

}else{

就添加class addClass()

}

②问题: 能不能用toggleClass完全代替前边的addClass和removeClass

答: 不能！addClass永远是添加class， removeClass永远是移除class，但是toggleClass没准: 一次添加，一次移除，再一次才是添加，再一次又是移除！

③总结: 只有确实在有和没有一个class之间来回切换时，才用toggleClass

(3). 示例: 双态按钮

4\_class.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">  <meta charset="UTF-8">  <title></title>  <style>  .btn {  padding: 5px 10px;  border-radius: 3px;  border: 1px solid #aaa;  outline: none;  }  /\* 按钮抬起时的默认样子 \*/  .up {  background: #fff;  color: #333;  }  /\* 按钮按下时的样子 \*/  .down {  background: #ddd;  color: #fff;  }  </style>  </head>  <body>  <button class="btn up">双态按钮</button>  <script src="js/jquery-1.11.3.js"></script>  <script>  //双态按钮: 让按钮的class在up和down之间切换  //DOM 4步  //1. 查找触发事件的元素  //本例中: 用户点按钮触发变化  $("button")  //2. 绑定事件处理函数  //本例中: 单击触发变化  .click(function(){  //3. 查找要修改的元素  //本例中: 要修改的就是当前按钮自己  //var $btn=$(this);  //4. 修改元素  //本例中: 如果有down class，就移除down class，否则如果没有down class，就添加down class  // if($btn.hasClass("down")){  // $btn.removeClass("down")  // }else{  // $btn.addClass("down");  // }  //其实就是在有和没有down这个class之间来回切换！  $(this).toggleClass("down");  })  </script>  </body>  </html>  运行结果: |

|  |
| --- |
| 简写: .attr()/.prop()/.css() 其实都可以一句话修改多个属性值  $元素.attr或prop或css({  属性名: 属性值,  ... : ...  }) |

三. 按节点间关系查找：

对应小程序视频: 小程序->在线->JQUERY->day02 6. jQuery按节点间关系查找

1. 父子关系:

(1). 获得一个元素的父元素: $元素.parent() 代替 元素.parentElement

(2). 获得一个元素下所有直接子元素: $元素.children() 代替 元素.children

a. 问题: children默认只能选择所有直接子元素，无法只选择个别想要的子元素

解决: $元素.children(选择器)， 意为可以有选择的挑选部分符合条件的元素

b. 问题: children默认只能在直接子元素中查找！无法在所有后代中查找

解决: $元素.find(选择器) 在所有后代元素中查找符合要求的元素

(3). 获得一个元素下第一个直接子元素: ~~元素.firstElementChild~~

$元素.children(":first-child");

(4). 获得一个元素下最后一个直接子元素: ~~元素.lastElementChild~~

$元素.children(":last-child");

2. 兄弟关系:

(1). 前一个兄弟:

a. DOM中: 元素.previousElementSibling

b. jq中: $元素.prev()

c. 问题: 有时，不止找前一个元素，想找之前所有元素

解决: $元素.prevAll(选择器)

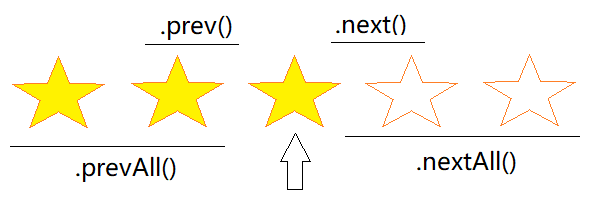
(2). 下一个兄弟:

a. DOM中： 元素.nextElementSibling

b. jq中: $元素.next();

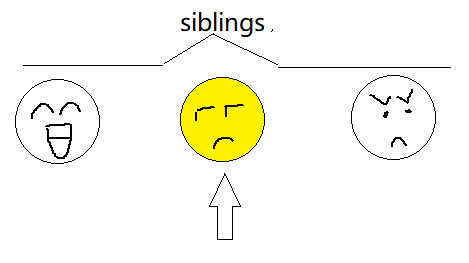
c. 问题: 有时，不止找下一个元素，想找之后所有元素

解决: $元素.nextAll(选择器)



(3). 问题: 有时，我们希望获得除当前元素之外其余所有兄弟元素（不论前后）！

解决: $元素.siblings("选择器")

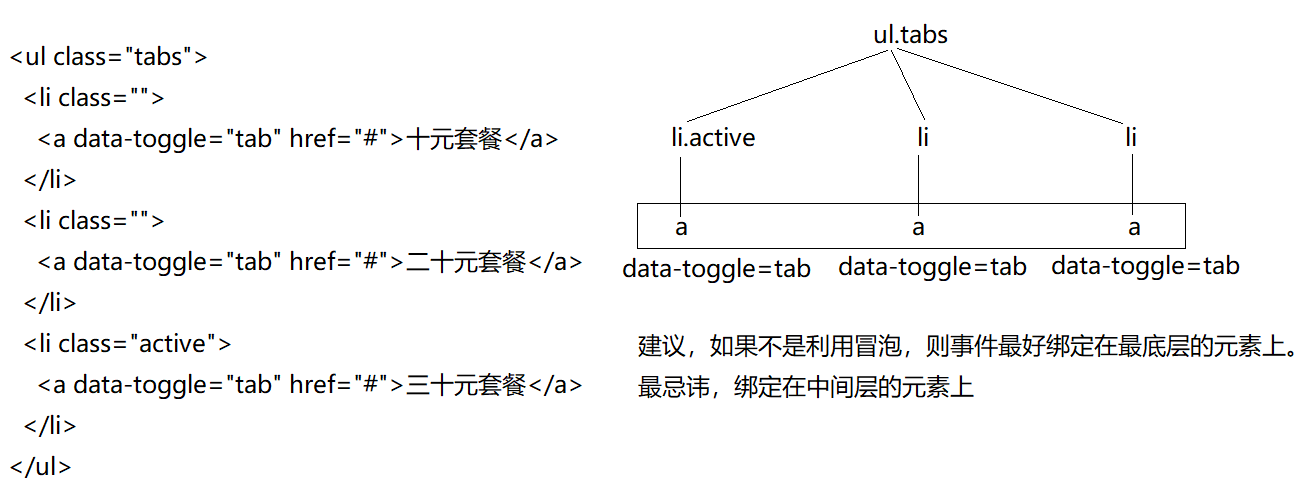


3. 示例: 使用按节点间关系查找，选择指定的元素

6\_traverse.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <title>...</title>  <script>  </script>  </head>  <body>  <!-- ul.top>(li.parent>ul>li.child\*3)\*2 -->  <ul class="top">  <li class="parent1">parent1  <ul>  <li class="child">child1</li>  <li class="child">child2</li>  <li class="child">child3</li>  </ul>  </li>  <li class="parent2">parent2  <ul>  <li class="child">child1</li>  <li class="child">child2</li>  <li class="child">child3</li>  </ul>  </li>  </ul>  <script src="js/jquery-1.11.3.js"></script>  <script>  //修改class为top的ul的所有直接子元素  $("ul.top").children().css("border","1px solid red")  //修改class为top的ul的所有后代li  $("ul.top").find("li").css("box-shadow","0 0 5px green")  //为class为child的li绑定单击事件  $("li.child").click(function(){  //选择当前元素的下一个元素/前一个元素/之前所有/之后所有/除自己之外所有  $(this).siblings()//.nextAll()//.prevAll()//.prev()//.next()  .css("background-color","yellow");  })  </script>  </body>  </html>  运行结果: |

4. 示例: 标签页效果



7\_tabs.html

|  |
| --- |
| <!doctype html>  <html lang="en">  <head>  <meta charset="UTF-8">  <style>  .tabs {  list-style: none;  padding: 0  }  .tabs a {  text-decoration: none;  color: #000;  padding: 6px 12px;  display: inline-block;  }  .tabs>li {  float: left;  border-bottom: 1px solid #000;  }  .tabs>.active {  border: 1px solid #000;  border-bottom: 0;  }  </style>  </head>  <body>  <h1>使用属性选择器实现标签页头的切换</h1>  <ul class="tabs">  <li class="active"><a data-toggle="tab" href="#">十元套餐</a></li>  <li><a data-toggle="tab" href="#">二十元套餐</a></li>  <li><a data-toggle="tab" href="#">三十元套餐</a></li>  </ul>  <script src="js/jquery-1.11.3.js"></script>  <script>  //DOM 4步  //1 查找触发事件的元素  //本例中: 用户点有data-toggle属性，且属性值为tab的元素触发变化  $("[data-toggle=tab]")  //2. 绑定事件处理函数  //本例中: 单击触发变化  .click(function(){  //3. 查找要修改的元素  //4. 修改元素  //本例中: 先给当前a的父元素li加active class，再去掉当前a的父元素里的兄弟身上的active class  $(this).parent()  // a 当前li  .addClass("active")//本来不需要返回值  //jQuery要求addClass必须返回.前的主语  //return $当前li  //addClass的返回值，刚巧和下一个操作的.前的主语一致！  .siblings().removeClass("active");  // 其它li  //jquery链式操作！  })  </script>  </body>  </html>  运行结果: |

四. 添加/删除/替换/克隆:

对应小程序视频: 小程序->在线->JQUERY->day02 7. jQuery添加删除元素

1. 添加:

(1). 回顾: DOM中: 3步:

a. 创建新元素: var 新元素=document.createElement("标签名")

b. 设置关键属性: 元素.属性名=属性值

c. 将新元素添加到DOM树: 3种:

1). 父元素末尾追加新元素: 父元素.appendChild(新元素)

2). 插入到父元素下一个现有子元素之前: 父元素.insertBefore(新元素, 现有子元素)

3). 替换父元素下一个现有子元素: 父元素.replaceChild(新元素, 现有子元素)

(2). jq中: 2步:

a. 用一段HTML片段创建一个或多个元素:

var $新元素=$(`HTML代码片段`);

问题: 依然是在内存中游离的，没有在DOM树上！所以，页面上暂时看不到

解决: 挂载

b. 将新元素添加到DOM树: 5种情况，10个函数

1). 父元素末尾追加新元素:

$父元素.append($新元素) 代替 父元素.appendChild(新元素)

兄弟函数: $新元素.appendTo($父元素)

2). 父元素开头插入新元素:

新增: $父元素.prepend($新元素)

兄弟函数: $新元素.prependTo($父元素)

3). 插入到一个现有子元素之前:

$现有元素.before($新元素) 代替 父元素.insertBefore(新元素, 现有子元素)

兄弟函数: $新元素.insertBefore($现有元素)

4). 插入到一个现有子元素之后:

新增: $现有元素.after($新元素)

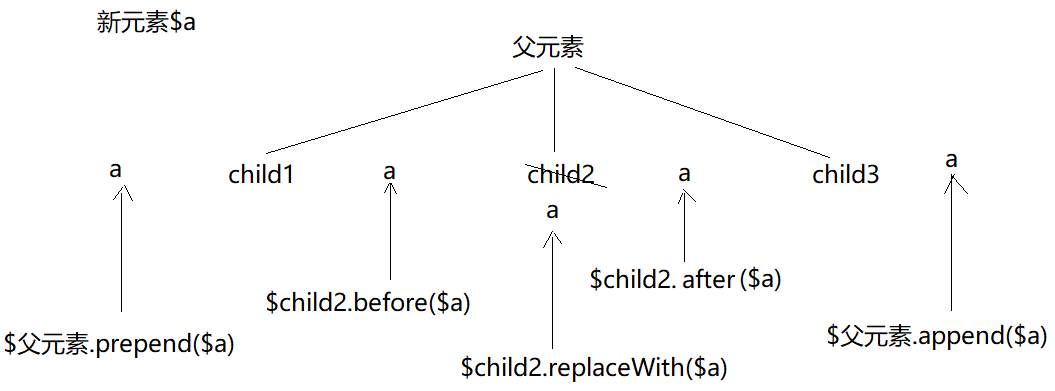
兄弟函数: $新元素.insertAfter($现有元素)

5). 替换父元素下一个现有子元素:

$现有元素.replaceWith($新元素) 父元素.replaceChild(新元素, 现有子元素)

兄弟函数: $新元素.replaceAll($现有元素)

|  |
| --- |
| 为什么每个函数又多出一个功能一模一样，只是主语和宾语颠倒位置的新函数？  因为主语不同，则函数返回的jquery对象不用。则后续可连接的链式操作也就不同！  如果希望在追加一个新元素后，继续对新元素执行下一步操作：  不好的做法: $父元素.append($新元素)  return $父元素，下一步用不了！  $新元素.下一步操作  好的做法: $新元素.appendTo($父元素).下一步操作()  return $新元素 下一步可继续使用 |



2. 删除: $元素.remove()

3. 示例: 点按钮，添加方块，点x删除方块

9\_append.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title> new document </title>  <meta charset="utf-8">  <style>  .container {  border: 1px solid #aaa;  overflow: hidden;  }  .block {  float: left;  margin: 10px;  border: 1px solid #aaa;  background: #faa;  width: 150px;  height: 150px;  }  .block:hover {  box-shadow: 0 5px 6px #000;  }  .close {  float: right;  padding: 5px;  font-weight: bold;  opacity: .2;  cursor: pointer;  }  .close:hover {  opacity: .5;  }  </style>  </head>  <body>  <h1>添加/删除节点</h1>  <button id="add-block">添加区块</button>  <div class="container">  <!-- <div class="block">  <span class="close">×</span>  </div> -->  </div>  <script src="js/jquery-1.11.3.js"></script>  <script>  //DOM 4步  //1. 查找触发事件的元素  //本例中: 点击按钮添加方块  $("#add-block")  //2. 绑定事件处理函数  .click(function(){  // //3. 查找要修改的元素  // //本例中: 向class="container"的div开头插入一个新方块div元素  // var $container=$(".container");  // //4. 修改元素  // //本例中:  // //先创建一个新的方块元素对象  // var $block=$(`<div class="block">  // <span class="close">×</span>  // </div>`);  // //设置方块的随机背景色  // $block.css("background-color",`rgb(${  // parseInt(Math.random()\*256)  // },${  // parseInt(Math.random()\*256)  // },${  // parseInt(Math.random()\*256)  // })`);  // //最后将方块插入到$container开头  // $container.prepend($block);  //创建一个新方块  $(`<div class="block">  <span class="close">×</span>  </div>`)  //设置方块的随机背景色  .css("background-color",`rgb(${  parseInt(Math.random()\*256)  },${  parseInt(Math.random()\*256)  },${  parseInt(Math.random()\*256)  })`)  //return 新方块  //将方块插入到父元素开头  //.prependTo($(".container"))  //简写: 不用自己找.container，直接把选择器给prependTo，prependTo会自动查找选择器所指的父元素  .prependTo(".container");  })    //实现点x按钮删除当前方块  //DOM 4步  //1. 查找触发事件的元素  //本例中: 因为多个x都可单击，所以应该用事件委托优化——复习事件委托  //事件应该只绑定在父元素上一份即可  $(".container")  //2. 绑定事件处理函数  //本例中: 应该先获得事件对象e，才能用e.target获得目标元素x  .click(function(e){  //获得当前点击的元素  var $target=$(e.target);  //判断当前实际点的元素是否是想要的  //本例中: 只有点在class为close的span元素上，才能删除  if($target.hasClass("close")){ //说明我们点的是x  //3. 查找要修改的元素  //本例中: 点x，其实是要删除x的父元素div  $target.parent()  //4. 修改元素  //本例中: 删除父元素div  .remove();  }  })    </script>  </body>  </html>  运行结果: |

4. 克隆: ...待续

总结: DOM操作万能4步:

1. 查找触发事件的元素

2. 绑定事件处理函数

3. 查找要修改的元素

4. 修改元素: 包含三步

取出旧值，修改后，将新值放回元素上

总结: $的原理: new jQuery() 4种用法:

1. $("选择器") 2件事:

(1). 创建jQuery子对象，

(2). 查找符合条件的DOM元素，保存进jQuery对象中

2. $(DOM元素) 创建jQuery子对象，并无需查找就可将已经获得的DOM元素保存进jQuery对象中.

3. $(`HTML片段`) 将HTML片段创建为新DOM元素对象

4. ?

总结: jQuery API 三大特点:

1. 自带遍历: 只对jQuery对象调用一次简化版函数，等效于自动对jQuery对象内保存的每个DOM元素都调用一次等效的原生函数或属性。

2. 一个函数两用: 和修改、事件绑定相关的函数

(1). 不传入新值时，默认执行获取旧值的操作

(2). 传入新值时，自动切换为执行修改操作

3. 凡是没有特定返回值的函数，几乎都返回.前正在操作的jQuery对象本身！

如果前一个函数正在操作的.前的属于和后一个操作想要的.前的主语，刚好一样！则可以使用链式操作，避免使用变量或重复代码！

总结: jQuery 不用背！！！用的时候来找到即可！

1. 查找元素:

(1). 按选择器查找: var $jquery对象=$("选择器")

a. jQuery支持所有CSS3选择器

b. 扩展:

1). 基本过滤: :first :last :even :odd :eq(i) :gt(i) :lt(i)

2). 内容过滤: :contains(关键词) :has(选择器) :parent :empty

3). 可见性过滤: :visible :hidden

4). 表单元素过滤:

i. :input 选择所有表单元素

ii. :text :password :button :radio ... ...

(2). 不需要查找就可直接获得的元素对象: 1个

a. $(document) 根节点对象

(3). 按节点间关系查找:

a. 父子关系:

1). $元素.parent() 代替 元素.parentElement  
 2). $元素.children("选择器") 代替 元素.children ——只在直接子元素中查找

新增: $元素.find("选择器") 在所有后代中查找

3). $元素.children(":first-child") 代替 元素.firstElementChild

4). $元素.children(":last-child") 代替 元素.lastElementChild

b. 兄弟关系: 2个属性

1). $元素.prev() 代替 元素.previousElementSibling

新增: $元素.prevAll("选择器")

2). $元素.next() 代替 元素.nextElementSibling

新增: $元素.nextAll("选择器")

3). 新增: 查找除当前元素之外其余所有兄弟: $元素.siblings("选择器")

2. 修改: 3种

(1). 内容: 3种

a. $元素.html() 代替了 .innerHTML

b. $元素.text() 代替了 .textContent

c. $元素.val() 代替了 .value

(2). 属性: 3种:

|  |  |  |
| --- | --- | --- |
|  | DOM | JQUERY |
| 字符串类型标准属性 | 核心DOM: getAttribute() setAttribute()  HTML DOM: 元素.属性名 | $元素.attr()  $元素.prop() |
| bool类型标准属性 | HTML DOM: 元素.属性名 | $元素.prop() |
| 自定义扩展属性 | 核心DOM: getAttribute() setAttribute()  HTML 5: 元素.dataset.自定义属性 | $元素.attr()  没有对应！ |

|  |  |  |
| --- | --- | --- |
|  | $元素.attr() | $元素.prop() |
| 字符串类型标准属性 | √ | √ |
| bool类型标准属性 | × | √ |
| 自定义扩展属性 | √ | × |

a. 字符串类型的HTML标准属性: 2种:

1). $元素.attr("属性名","属性值") 代替: getAttribute() setAttribute()

2). $元素.prop("属性名","属性值") 代替: 元素.属性名

b. bool类型的HTML标准属性: 只能用 $元素.prop("属性名",bool)

c. 自定义扩展属性: 1种

1). $元素.attr("data-自定义属性名","属性值")

代替 元素.getAttribute() 元素.setAttribute()

3). 查找带有自定义扩展属性的元素: 属性选择器 [data-自定义属性=属性值]

(3). 样式:

a. 只获取或修改一个css属性时: 统一为: $元素.css("css属性名", "属性值")

b. 批量修改多个css属性时: 用class

1). $元素.addClass()

2). $元素.removeClass()

3). $元素.hasClass()

4). $元素.toggleClass();

3. 添加元素:

(1). 添加元素: jquery只需要2步

a. 用HTML片段创建新元素

var $新元素=$(`HTML片段`)

说明: 元素所需的属性和样式已经包含在HTML片段里了，所以，无需再专门添加关键属性。

b. 将新元素添加到DOM树: 5种 10个函数

1). 父元素末尾追加新元素:

$父元素.append($新元素) 代替 父元素.appendChild(新元素)

兄弟函数: $新元素.appendTo($父元素)

2). 父元素开头插入新元素:

新增: $父元素.prepend($新元素)

兄弟函数: $新元素.prependTo($父元素)

3). 插入到一个现有子元素之前:

$现有元素.before($新元素) 代替 父元素.insertBefore(新元素, 现有子元素)

兄弟函数: $新元素.insertBefore($现有元素)

4). 插入到一个现有子元素之后:

新增: $现有元素.after($新元素)

兄弟函数: $新元素.insertAfter($现有元素)

5). 替换父元素下一个现有子元素:

$现有元素.replaceWith($新元素) 父元素.replaceChild(新元素, 现有子元素)

兄弟函数: $新元素.replaceAll($现有元素)

(2). jQuery内已封装了尽量减少操作DOM树的优化(文档片段)，所以我们无需额外专门创建文档片段，jquery内部也已经自动使用了文档片段！

4. 删除元素: $元素.remove() 代替 父元素.removeChild(要删除的元素)

5. 事件:

(1). 事件绑定: 3种:

a. <元素 on事件名="事件处理函数()">

function 事件处理函数(){ ... }

b. 一个元素的一个事件属性上，只绑定一个事件处理函数，不会动态增加或减少:

元素.on事件名=function(){ ... }

c. 一个元素的一个事件上，需要同时绑定多个事件处理函数，还可能动态添加或移除

元素.addEventListener("事件名", 事件处理函数)

元素.removeEventListener("事件名", 原事件处理函数)

(2). 事件模型: 3个阶段: 捕获、目标触发、冒泡

(3). 获得事件对象: 定义事件绑定时: 元素.on事件名=function(e){ //触发事件时e=event }

(4). 取消冒泡/停止蔓延: e.stopPropagation()

(5). 只要多个平级子元素都要绑定相同的事件时: 利用冒泡/事件委托: 3步

a. 事件绑定在父元素上一份

b. 处理函数中，用e.target代替this获得实际点击的那个子元素

c. 处理函数中，先验证e.target的特征是想要的元素，才继续执行后续操作。

(6). 阻止元素自带的默认行为: e.preventDefault();

(7). 获得鼠标位置：

a. 鼠标位置相对屏幕左上角的距离: e.screenX e.screenY

b. 鼠标位置相对浏览器文档显示区左上角的距离 e.clientX e.clientY

c. 鼠标位置相对事件所在元素左上角的距离e.offsetX e.offsetY

(8). 窗口滚动事件:

a. 当滚动条滚动时自动触发: window.onscroll=function(){ ... }

b. 获取滚动条滚动过的距离:

var scrollTop=document.body.scrollTop||document.documentElement.scrollTop;

c. 控制滚动条滚动到指定位置: window.scrollTo(0, 要滚动到的位置)

总结: this 6种:

1. obj.fun() this->obj

2. new Fun() this->new正在创建的子对象

3. 类型名.prototype.共有方法=function(){ ... } this->将来调用这个共有方法的.前的子对象

4. fun() 和 (function(){ })() 和回调函数中的this->window

5. 事件处理函数中的this->当前正在触发事件的这个元素对象

6. $jquery对象.click(function(){ this->将来点击的那个DOM元素对象})

今日对应小程序视频:

小程序->在线->JQUERY->day01 4. jQuery其它选择器...

小程序->在线->JQUERY->day02 1. jquery 修改元素的内容(有扩展)

2. jquery修改元素属性attr() prop()

3. jquery修改样式css()...

6. jquery按节点间关系查找

7. jQuery添加删除元素

作业:

1. 复习今日小程序问题清单(小程序->在线->JQUERY->day01、02)，预习明日小程序问题清单(小程序->在线->jQuery->day03)

2. 看小程序视频预习jQuery事件绑定和事件委托:

小程序->在线->JQUERY->day02 12 jquery事件 事件委托

3.（凡是觉得知识点串不起来）看小程序视频 完成 DOM版学子商城项目

小程序->在线->DOM->day04 扩展: DOM版学子商城...

家里环境xampp和nodejs能用，且第一阶段学的好的同学:

从小程序中网盘地址中下载xzserver\_start.zip，看视频中day01和day02，跟着做

家里环境xampp和nodejs不能用，或者第一阶段学的不好的同学：

从小程序中网盘地址中只下载public.zip，看视频中day01和day02，跟着做

然后，所有网页用live server运行

视频中所有ajax请求的服务器端接口地址，都改为我的新浪云服务器地址，就不需要你再运行服务器端了。也不需要你了解任何服务器端知识了:

/index 改为 http://xzserver.applinzi.com/index

/details 改为 http://xzserver.applinzi.com/details?lid=xxx