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-> day02、day03

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-> day02、day03

(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

12. jQuery事件 事件委托

小程序->在线->JQUERY->day03

作业: jquery动画函数 手风琴效果

作业: 高频笔试题！防抖和节流

3. 今日课程中包含的英语单词的翻译：小程序->单词->JQUERY->day02、03

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7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. 添加/删除/替换/克隆

2. 事件

3. 动画

一. 添加/删除/替换/克隆

对应小程序视频: 小程序->在线->JQUERY->day02 7. 添加 删除 替换 克隆

1. 添加: 2步

(1). 创建新元素: var $新元素=$(`HTML片段`)

(2). 将新元素添加到DOM树:

a. 末尾追加  
 b. 开头插入

c. 插入到现有元素前

d. 插入到现有元素后

e. 替换

2. 删除: $元素.remove();

3. 克隆:

(1). 什么是: 复制出一个一模一样的新的元素对象

(2). 如何: var $新元素副本=$原元素.clone()

(3). 示例: 选择主角飞机

day02 clone/10\_replace\_clone.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">  <meta charset="UTF-8">  <title></title>  <style>  body {  text-align: center;  }  </style>  </head>  <body>  <h1>替换节点</h1>  <div id="chosen">  <img src="img/p0.png">  </div>  <hr />  <div id="list">  <img src="img/p3.png">  <img src="img/p4.png">  <img src="img/p5.png">  </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: id为list下的所有img都可点击  //所以，应该用事件委托优化，事件应该只绑定在父元素上一份  $("#list")  //2. 绑定事件处理函数  //本例中: 单击图片触发变化  .click(function(e){  //用e.target代替this获得当前点击的img元素  var $target=$(e.target);  //验证$target是不是想要的  //本例中: 只有点在img元素上才能触发变化  //if($target是img){  //错误: jQuery是函数库，不能使用.属性  //if($target.nodeName=="IMG"){  //正确: jQuery用.prop()代替了.属性  //if($target.prop("nodeName")=="IMG"){  //新增: $元素.is("选择器")  // 元素选择器  if($target.is("img")){  // //3. 查找要修改的元素  // //本例中: 点击下方飞机图片，应该先将当前img复制一个副本，用副本去代替id为chosen的div下的img  // var $chosen=$("#chosen>img");  // //4. 修改元素  // $target.clone().replaceAll($chosen);  // //原img 副本img  //简写:  $target.clone().replaceAll("#chosen>img") //不用自己找，replaceAll自带查找功能！  }  })  </script>  </body>  </html>  运行结果: |

|  |
| --- |
| 新增:  var bool=$元素.is("选择器") 专门判断一个元素是否符合选择器的要求 |

二. 事件:

对应小程序视频: 小程序->在线->JQUERY->day02 8. jquery事件 事件委托 on off

1. 绑定事件:

(1). 回顾: DOM中: js中2种:

~~a. 赋值方式: 元素.on事件名=function(e){~~

this->当前正在触发事件的这个元素

}

b. 添加事件监听: 元素.addEventListener("事件名",事件处理函数)

移除事件监听: 元素.removeEventListener("事件名", 原事件处理函数)

(2). jq中: 1种

a. 标准写法: $元素.on("事件名", 事件处理函数)

$元素.off("事件名", 原事件处理函数)

b. 特殊:

1). DOM中: 有名的事件处理函数，只能添加一次，不能反复添加多次

2). jq中: 因为jQuery底层的特殊处理，所以$元素.on()即使使用有名称的函数，也可反复添加事件处理函数。移除时，用函数名可移除添加的多个同名事件处理函数

c. 简写: jQuery对十几种最常用事件，提供了更简化的简写方式

1). $元素.事件名(function(e){

this->当前正在触发事件的DOM元素

})

2). 比如: $元素.click(...) $元素.blur(...) ... ...

d. 总结: 哪些事件可简写:

|  |
| --- |
| 常用事件名:  blur 失去焦点  change 下拉列表选中项改变  click 单击  dblclick 双击  focus 获得焦点  keydown 键盘按键按下  keyup 键盘按键抬起  mousedown 鼠标按键按下  mouseenter 鼠标进入(jq)  mouseleave 鼠标移出(jq)  mousemove 鼠标移动  mouseout 鼠标移出(dom)  mouseover 鼠标进入(dom)  mouseup 鼠标按键抬起  resize 窗口大小改变  scroll 网页滚动 |

e. 其实:

~~$元素.click() 底层 自动翻译为 元素.onclick=function(){ ... }~~

$元素.click() 底层 自动翻译为 元素.addEventListener("click", ...)

(3). 示例: 点击按钮 发射普通子弹，点击获得奖励按钮，获得跟踪导弹，还可移除导弹

event/12\_bind\_unbind.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title> new document </title>  <meta charset="utf-8">  </head>  <body>  <h1>事件绑定</h1>  <button id="btn1">发射子弹</button>  <button id="btn2">获得奖励</button>  <button id="btn3">失去奖励</button>  <script src="js/jquery-1.11.3.js"></script>  <script>  //开局: 点击btn1，发射普通子弹  //btn1.addEventListener("click",function(){ ... })  // ||  //$("#btn1").on("click",function(){  // ||  $("#btn1").click(function(){  console.log(`发射普通子弹......`)  })  //因为发射导弹的处理函数有可能被移除，所以应该用有名称的函数  function shoot2(){  alert("发射跟踪导弹=>=>=>")  }  //点btn2，可以给btn1多添加一种跟踪导弹  //$("#btn2").on("click",function(){  // ||  $("#btn2").click(function(){  //btn1.addEventListener("click",shoot2)  // ||  //$("#btn1").on("click",shoot2)  // ||  $("#btn1").click(shoot2)  })  //点btn3, 可以从btn1上移除跟踪导弹  //$("#btn3").on("click",function(){  // ||  $("#btn3").click(function(){  //btn1.removeEventListener("click",shoot2)  // ||  $("#btn1").off("click",shoot2)  })  </script>  </body>  </html>  运行结果： |

2.事件委托：

(1). 何时: 多个平级子元素要绑定相同的事件时

(2). DOM中: 3步 —— 对于今后框架开发才是重要的！

a. 事件只绑定在父元素上一份即可

b. 用e.target代替this，来获得最初点击的目标元素

c. 判断e.target的特征是想要的，才继续执行后续操作

(3). jq中: 其实也可以用DOM版的事件委托，但是jQuery提供了事件委托的简化

a. 事件只绑定在父元素上一份即可，但是必须用.on()绑定

b. 不再用e.target代替this，jquery又让this重新指向了当初触发事件的那个子元素 ——又可以用this了

c. 不用自己写if来判断点击的元素是否想要的。而是在.on()中添加第二个选择器参数。由on()自动用选择器先验证当前点击的元素是否想要的，然后再决定是否执行事件处理函数

d. 总结:

$父元素.on("事件名", "选择器", function(){

//this->最初点击的那个子元素，而不再指父元素了！

})

e. 但是, 这种简写只有jQuery中能用！DOM中或今后的其它框架中都不能用！所以，这个简写了解即可！

(4). 示例: 选择飞机

day02 clone/10\_replace\_clone\_delegate.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">  <meta charset="UTF-8">  <title></title>  <style>  body {  text-align: center;  }  </style>  </head>  <body>  <h1>替换节点</h1>  <div id="chosen">  <img src="img/p0.png">  </div>  <hr />  <div id="list">  <img src="img/p3.png">  <img src="img/p4.png">  <img src="img/p5.png">  </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: id为list下的所有img都可点击  //1. 所以，应该用事件委托优化，事件应该只绑定在父元素上一份  $("#list")  //2. 绑定事件处理函数  //本例中: 单击图片触发变化  //只有点在img元素上才能触发变化  // 元素选择器  .on("click","img",function(){  //3. 查找要修改的元素  //4. 修改元素  //本例中: 先将当前点中的img克隆一个一模一样的副本，用副本代替id为chosen的div下的img。  $(this).clone().replaceAll("#chosen>img") //不用自己找，replaceAll自带查找功能！  })  </script>  </body>  </html>  运行结果: |

3. 页面加载后自动执行:

对应小程序视频: 小程序->在线->JQUERY->day03 1. 页面加载完成后自动执行...

(1). 问题1: 将来我们的js都是写在网页外部的独立的js文件中的。通过<script>标签引入到网页中。应该在网页开头的<head>中引入，还是在<body>结尾引入？

(2). 答: 如果希望页面一加载就自动执行的js，应该放在<body>结尾引入！因为js中通常都要操作HTML中的元素。必须先有HTML，才能执行js操作HTML。所以js必须在要操作的HTML之后才行！

(3). 问题2: 公司里什么人都有，有人偏偏就把js在<head>中引入！如何保证js文件无论在开头引入，还是在结尾引入，都能注定在页面加载完成后才自动执行？

(4). 解决

a. 其实整个网页加载完成时，浏览器会自动触发一个load事件。

放在load事件中的代码，默认注定会在整个页面加载完成后，才自动执行！

b. 所以: window.onload=function(){

//这里的代码，无论何时定义，注定只会在整个页面所有内容加载完之后才执行！

}

(5). 问题3: window.onload=function(){}必须等待所有页面内容(HTML+JS+CSS+图片)全部下载完成，才能触发，才能绑定事件，让用户用上功能——晚。有些功能用户根本不关心界面的css和图片是否好看！一心只想第一时间赶紧用上功能！——抢票

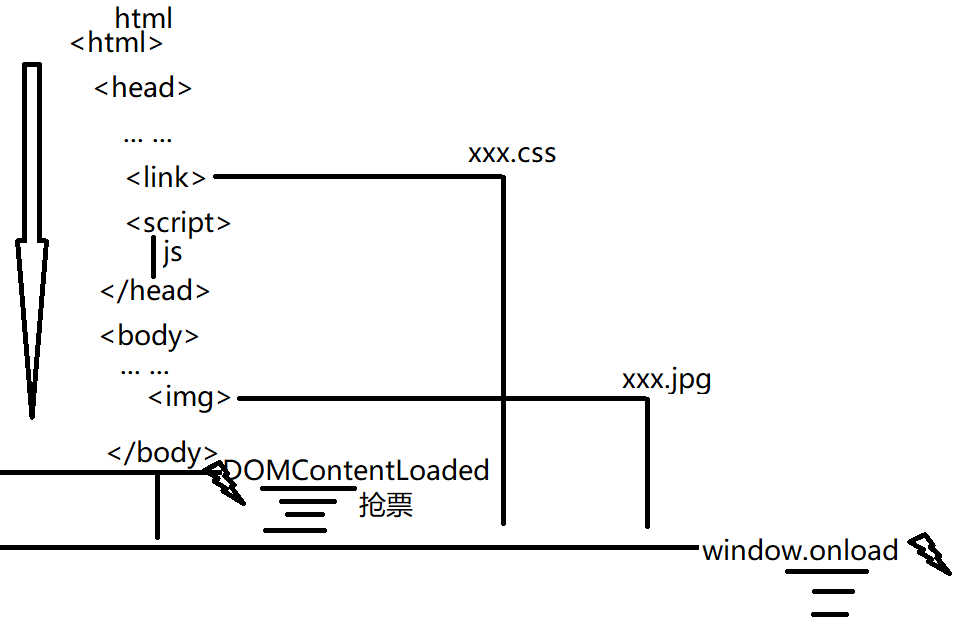
(6). 解决:

a. 其实在window.onload之前，仅HTML和js下载完成后，就提前有一个事件触发:

DOMContentLoaded 仅DOM内容加载完成

放在这个事件中的代码，会比window.onload触发早的多

因为不用等大量的css和图片下载完，就可提前触发



b. 问题: DOMContentLoaded有兼容性问题

c. 解决: jQuery中提供了一个没有兼容性问题的统一做法代替DOMContentLoaded

DOM内容 就绪

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

(7). 问题: $(document).ready(太长了！

(8). 解决: 今后绑定仅DOM内容加载完成可简写为：

$(function(){ ... })

(9). 总结: 今后所有不依赖于css和图片的大部分页面初始化jquery代码，都应该放在$(function(){ ... })中！

(10). $(function(){ ... }) 和(function(){ ... })()

a. (function(){ ... })() 匿名函数自调 立刻执行其中的代码，执行后立刻释放

b. $(function(){ ... }) 事件绑定 不是立刻执行，必须等到网页中HTML+JS都加载完成了，才自动执行

c. 为了避免使用全局变量，js代码应该包含在匿名函数自调中。但是进入jquery之后，$(function(){ ... })也可以将变量限制在function内部，也可以避免使用全局变量。所以，jquery中，写了$(function(){ ... })，就~~不用重复写(function(){ ... })()~~

(11). 示例: 仅DOM内容加载完成，就提前给按钮绑定单击事件

event/14\_ready.js

|  |
| --- |
| //希望仅DOM内容（HTML+JS）加载完就提前出发  //固定套路！  //DOMContentLoaded  //$(document).ready(function(){ //早  //||  $(function(){  alert("仅DOM内容（HTML+JS）加载完")  //自动给id为btn的按钮，绑定单击事件  $("#btn").click(function(){  alert("抢票！")  })  });  //希望当整个页面加载完成后，才  window.onload=function(){ //晚  alert("整个页面加载完成！")  } |

event/14\_ready.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title> new document </title>  <meta charset="utf-8">  <script src="js/jquery-1.11.3.js"></script>  <script src="14\_ready.js">  //$(function(){  //$("#btn").click(function(){ ... })  //});  //window.onload=function(){ ... }  </script>  </head>  <body>  <button id="btn">click me</button>  </body>  </html>  <!--仅DOM内容: HTML+JS 加载完先触发DOMContentLoaded，也就是$(function(){...}) 此时，页面中已经有HTML内容，所有事件可以正常绑定到元素上-->  <!--然后,当所有页面内容(HTML+JS+CSS+图片)加载完成后才自动执行window.onload() 稍晚一些-->  运行结果: |

4. 鼠标事件:

对应小程序视频: 小程序->在线->JQUERY->day-3 2. 鼠标事件 模拟触发

(1). 回顾: DOM中:

a. mouseover: 当鼠标进入某个元素范围内时自动触发

鼠标 进入

b. mouseout: 当鼠标离开某个元素范围时自动触发

鼠标 离开

(2). 问题: 反复进出子元素，也会反复冒泡触发父元素上的鼠标进出事件。和现实不符！

(3). 解决: jquery中或将来框架中都新增mouseenter(鼠标进入)和mouseleave(鼠标离开)代替mouseover和mouseout

(4). 优点: 反复进出子元素，不会反复触发父元素上的进出事件，和现实相符。

(5). 今后jquery中或框架中如果能用mouseenter和mouseleave，都首选这一组！

(6). 示例: 使用mouseenter和mouseleave代替mouseover和mouseout

event/15\_mouse.html

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>  <title>事件处理</title>  <meta charset="utf-8" />  <style>  #d1 #d2 #d3 {  cursor: pointer  }  #d1 {  background-color: green;  position: relative;  width: 150px;  height: 150px;  text-align: center;  cursor: pointer;  }  #d2 {  background-color: blue;  position: absolute;  top: 25px;  left: 25px;  width: 100px;  height: 100px;  }  #d3 {  background-color: red;  position: absolute;  top: 25px;  left: 25px;  width: 50px;  height: 50px;  line-height: 50px;  }  </style>  </head>  <body>  <div id="d1">  <div id="d2">  <div id="d3">  </div>  </div>  </div>  <script src="js/jquery-1.11.3.js"></script>  <script>  //当鼠标进入最外层的绿色d1范围内时，提示进入d1  //当鼠标离开最外层的绿色d1范围时，提示离开d1  $("#d1")  // //.mouseover(function(){  // .mouseenter(function(){  // console.log(`鼠标进入d1`)  // })//return $("#d1")  // //.mouseout(function(){  // .mouseleave(function(){  // console.log(`鼠标离开d1`)  // })  .hover(  function(){ console.log(`鼠标进入d1`) },//给mouseenter  function(){ console.log(`鼠标离开d1`) } //给mouseleave  )  </script>  </body>  </html>  运行结果： |

(7). 简写: 很多情况下都会同时绑定鼠标进入和鼠标移出事件

a. 今后，只要同时绑定鼠标进入和鼠标移出事件时，只要绑定一个hover()就够了。

b. 如何: $元素.hover( //=mouseenter+mouseleave

function(){ ... }, //给mouseenter

function(){ ... } //给mouseleave

)

c. 简写: 如果可以将两个function通过特殊的手段改成一样的函数，则只写一个function即可！——少见，特例

$元素.hover(//=mouseenter+mouseleave

function(){ ... } //既给mouseenter，又给mouseleave

)

d. 强调: .hover()中只写一个函数时，~~不是只绑定mouseenter的意思！~~而也是同时绑定mouseenter和mouseleave的意思，且唯一的一个函数，既给mouseenter，又给mouseleave

e. 示例: 使用hover简化mouseenter和mouseleave

event/16\_hover.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title> new document </title>  <meta charset="utf-8">  <style>  #target {  border: 1px solid #eee;  border-radius: 6px;  padding: 10px;  transition: all .5s linear;  }  /\* 当鼠标进入div时，添加hover class，当鼠标移出div时，移除div的class hover \*/  #target.hover {  border: 1px solid #aaa;  box-shadow: 0 0 6px #aaa;  background-color:red;  color:#fff;  }  /\* 其实也可以用纯css做 \*/  /\* #target:hover {  border: 1px solid #aaa;  box-shadow: 0 0 6px #aaa;  background-color:red;  color:#fff;  } \*/  </style>  </head>  <body>  <h1>使用hover(fn,fn)</h1>  <h3>鼠标悬停在div上方，则突出显示；移出则取消突出显示</h3>  <div id="target">  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eligendi neque quae voluptatum ducimus culpa itaque maxime distinctio soluta cum cupiditate esse tenetur deserunt fuga perferendis sed veritatis asperiores. Numquam officia.</p>  </div>  <script src="js/jquery-1.11.3.js"></script>  <script>  // $("#target").hover( //=mouseenter+mouseleave  // function(){ $(this).addClass("hover") }, //给mouseenter  // function(){ $(this).removeClass("hover") } //给mouseleave  // )  $("#target").hover( //=mouseenter+mouseleave  function(){ $(this).toggleClass("hover") }, //既给mouseenter，又给mouseleave  )  </script>  </body>  </html>  运行结果: |

5. 模拟触发:

对应小程序视频: 小程序->在线->JQUERY->day-3 2. 鼠标事件 模拟触发

(1). 什么是: 不点击元素，也能触发元素上的事件处理函数

(2). 比如: 搜索时:

a. 点搜索按钮执行搜索

b. 在文本框中输入关键词，按回车也能实现和点搜索按钮完全一样的搜索功能

(3). 如何: $元素.trigger("事件名")

触发

(4). 意为: 找到指定元素，并触发元素上指定事件上绑定的事件处理函数

(5). 简写: 如果要触发的事件，属于十几种常用事件之一，可简写为:

$元素.事件名()

(6). 示例: 按回车也能搜索

event/17\_trigger.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">  <meta charset="UTF-8">  <title></title>  <style>  </style>  </head>  <body>  <input><button>百度一下</button>  <script src="js/jquery-1.11.3.js"></script>  <script>  //希望点按钮获取文本款中的关键词，执行搜索操作  //DOM 4步  //1. 查找触发事件的元素  //本例中: 点按钮执行搜索  $("button")  //2. 绑定事件处理函数  //本例中: 绑定单击事件  .click(function(){  //先获取文本框中的内容  var value=$(":text").val().trim();  //如果文本框中内容不为""  if(value!==""){  //再执行搜索操作  console.log(`查找 ${value} 相关的内容...`)  }  });  //希望在文本框上按回车,可执行和点击按钮一样的搜索功能  // 键盘按下再抬起  $(":text").keyup(function(e){  //只有按的是回车键时才执行搜索  //事件对象e中保存了一个keyCode属性，专门获取本次按键的序号  //如果按的是13号键，说明本次按的是回车  if(e.keyCode==13){  //模拟触发按钮的单击事件处理函数  //$("button").trigger("click")  //因为click事件属于常用事件列表  $("button").click();  }  })  </script>  </body>  </html>  运行结果： |

三. 动画: 2大类

1. 简单动画: 固定写死的三种动画效果，9个函数 —— 几乎不用！

(1). 显示隐藏: $元素.show() $元素.hide() $元素.toggle()

显示 隐藏 在显示和隐藏之间来回切换

display:block display:none

a. 问题: show()/hide()/toggle()不加任何参数，默认瞬间显示隐藏，没有动画效果

b. 解决: 只要给show()/hide()/toggle()加上一个动画持续时间毫秒数参数，就能有过渡

(2). 上滑下滑: $元素.slideUp() $元素.slideDown() $元素.slideToggle()

(3). 淡入淡出: $元素.fadeIn() $元素.fadeOut() $元素.fadeToggle()

|  |
| --- |
| 简单动画9个函数两个致命缺点:  1. 效果好丑，但是如果想改，不知道改哪儿！——几乎不可维护！  2. 在F12 element中看到变化过程！—— 说明这组动画使用js程序+定时器模拟的。——效率低！远不如css transition  总结: 今后只要简单动画效果，首选css transition做！  优点: 1. 便于维护！2. 比js程序效率高！ |

(4). 示例: 简单测试简单动画效果

1\_show\_hide.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title> new document </title>  <meta charset="utf-8">  </head>  <body>  <h1>jQuery动画函数——显示隐藏动画</h1>  <button id="btn1">显示/隐藏div</button>  <div id="target">  <p><span>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Tempore debitis animi sint iste sequi sunt ad  excepturi error labore molestiae est expedita eos nisi placeat provident dolorem quos facilis!  Sapiente!</span><span>Accusamus neque id reprehenderit! Voluptatem in deleniti laboriosam commodi facere magnam  impedit minima corrupti distinctio culpa amet optio natus esse. Inventore incidunt ab id perspiciatis atque  minus magnam tempore harum.</span></p>  </div>  <script src="js/jquery-1.11.3.js"></script>  <script>  $("#btn1").click(function () {  var $div=$("#target");  //如果div是隐藏的，就显示div  // if($div.is(":hidden")){  // $div.show();  // }else{//否则如果div是显示的，就隐藏div  // $div.hide();  // }  //$div.toggle(2000);  //$div.slideToggle();  $div.fadeToggle();  });  </script>  </body>  </html>  运行结果： |

(5). 如果不考虑过渡效果，show()/hide()/toggle()还是非常好用的！大大的简化了display:none和display:block！——没有性能损失，也不用考虑可维护性！——建议使用的！

(6). 示例: 切换精简显示品牌列表和显示全部品牌

2\_show\_hide2.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">  <meta charset="UTF-8">  <title></title>  <style>  body {  text-align: center;  }  ul {  list-style: none;  }  li {  margin: 0;  padding: 0;  display: inline-block;  width: 30%;  }  </style>  </head>  <body>  <ul id="list">  <li>尼康(234)</li>  <li>佳能(22)</li>  <li>索尼(932)</li>  <li>宾得(11)</li>  <li>爱国者(3234)</li>  <li>欧巴(32)</li>  <li>海鸥(2334)</li>  <li>卡西欧(334)</li>  <li>三星(834)</li>  <li>松下(234)</li>  <li>其它品牌(2343)</li>  </ul>  <button data-toggle="brandlist">精简显示品牌</button>  <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data-toggle="brandlist"的按钮  $("[data-toggle='brandlist']")  //2. 绑定事件处理函数  .click(function(){  //3. 查找要修改的元素  //本例中: 要修改ul下大于4位置且不是最后一个的li  $("ul>li:gt(4):not(:last)")  //4. 修改元素  //本例中: 切换这几个li的显示隐藏  .toggle();//在display:none和display:block之间来回切换  //修改当前按钮的文字:  var $btn=$(this);  //如果当前按钮的文字是: 精简显示品牌，就改为显示全部品牌  //否则如果当前按钮的文字是显示全部品牌，就改为精简显示品牌  if($btn.html()=="精简显示品牌"){  $btn.html("显示全部品牌");  }else{  $btn.html("精简显示品牌");  }  })  </script>  </body>  </html>  运行结果: |

2. 万能动画:

(1). 什么是: 能对大多数css属性应用动画效果的函数

(2). 如何: $元素.animate({

css属性: 目标值

}, 动画持续时间)

(3). 原理: animate函数会自动获得元素现在的状态和属性值，根据当前值与目标值之间的差距，自动计算动画变化的过程，并执行动画播放效果

(4). 问题: 也是用js程序+定时器模拟的——效率不如css

(5). 但是，animate毕竟比较灵活！所以，使用比较广泛！

(6). 问题: animate只支持单个数值的css属性。

不支持颜色动画和CSS3变换transform

但是CSS transition支持颜色动画和css3变换的！

3. 排队和并发:

(1). 并发: 放在一个animate中的多个css属性是同时变化的

(2). 排队: 多个css属性先后依次变化

a. 如何: 对一个元素先后调用多个animate()函数，多个animate()函数中的css属性是先后依次变化的！

b. 原理:

1). 每个元素身上都有一个动画队列

2). animate()其实并不是立刻执行动画的意思，仅仅是将动画加入到元素的动画队列中

3). 如果元素的动画队列中，没有正在播放的动画，则新加入的动画会立刻执行

4). 如果元素的动画队列中，有正在播放的动画，则新加入的动画，必须等待前一个动画播放完毕，才能开始播放！

4. 停止动画:

(1). $元素.stop()

(2). 坑: stop()默认只能停止队列中正在播放的一个动画。后续动画依然继续执行！

(3). 解决: $元素.stop(true) //既停止当前正在播放的动画，又清空后续队列

5. jQuery新增选择器: :animated 专门选择正在播放动画的元素

6. 动画结束后自动执行:

(1). 问题: animate()底层是定时器模拟的，所以animate()是异步的！放在animate()之后的代码不会等animate()执行完才执行，而是几乎和animate()同时执行。

(2). 解决: 所有动画函数都有最后一个回调函数参数，放在最后一个参数回调函数中的代码，注定会在动画播放结束后才执行

$元素.animate({ ... }, 持续时间, function(){ ... })

7. 示例: 四颗小星星，点每个小星星播放规定的动画效果

5\_stars.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">  <meta charset="UTF-8">  <title></title>  <style>  img {  position: relative;  }  </style>  </head>  <body>  <img id="s1" src="img/star.png"><br />  <img id="s2" src="img/star.png"><br />  <img id="s3" src="img/star.png"><br />  <img id="s4" src="img/star.png"><br />  <script src="js/jquery-1.11.3.js"></script>  <script>  /\*  s1在屏幕左上角的小星星， 点击后从左移动到屏幕右边  s2在屏幕左上角的小星星，点击后从左移动到屏幕右边，再移动到下边——走直角  s3在屏幕左上角的小星星，点击后从左上角移动到屏幕右下边，走斜线  s4点击小星星，变大、变淡.... 直至消失  \*/  $("#s1").click(function () {  var $s = $(this);  //希望点一次启动，再点一次停止  //如果当前星星正在播放动画，则停止动画  if ($s.is(":animated")) {  $s.stop();  } else {//否则如果当前星星没有播放动画，则启动动画  $s.animate({  left: 400  }, 3000)  }  })  $("#s2").click(function () {  var $s = $(this);  if($s.is(":animated")){  $s.stop(true)  }else{  $s.animate({  left: 400  }, 2000)  .animate({  top: 200  }, 1000)  }  })  $("#s3").click(function () {  var $s = $(this);  if($s.is(":animated")){  $s.stop()  }else{  $s.animate({  left: 400,  top: 200  }, 3000)  }  })  $("#s4").click(function(){  alert("疼！");  var $s=$(this)  $s.animate(  {//异步  width:256,  height:256,  opacity:0  },  2000,  function(){ //回到函数 注定会在动画播放结束后自动执行  //animate底层让this->animate.前的jq对象  $s.hide();//display:none  }  )    })  </script>  </body>  </html>  运行结果： |

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. $(function(){ ... }) 绑定DOMContentLoaded事件，让代码在HTML+JS加载完就可提前执行。

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: 增删改查+事件绑定 不用背！！！用的时候来找到即可！

1. 查找元素:

(1). 将所有的查找统一为按选择器查找: 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(input select textarea button)

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:

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:

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

|  |  |  |
| --- | --- | --- |
|  | 凡是字符串类型的属性 | 凡是标准属性 |
|  | $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();

|  |
| --- |
| 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,  ... : ...  }) |

3. 添加元素: jquery只需要2步

(1). 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。

(2). 将新元素添加到DOM树: 5种 10个函数

a. 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)

b. 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)

c. 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)

d. 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)

e. 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. 新增: 克隆: $元素.clone()

6. 新增: 判断元素是否符合选择器要求: $元素.is("选择器")

7. 事件:

(1). 事件绑定: 1种:

a. $元素.on("事件名", 事件处理函数) 代替 元素.addEventListener("事件名", 处理函数)

$元素.off("事件名", 原事件处理函数) 代替 元素.removeEventListener("事件名",原处理函数)

b. 简写: 只有以下列表中的事件才能: $元素.事件名(事件处理函数)

|  |
| --- |
| 常用事件名:  blur 失去焦点  change 下拉列表选中项改变  click 单击  dblclick 双击  focus 获得焦点  keydown 键盘按键按下  keyup 键盘按键抬起  mousedown 鼠标按键按下  mouseenter 鼠标进入(jq)  mouseleave 鼠标移出(jq)  mousemove 鼠标移动  mouseout 鼠标移出(dom)  mouseover 鼠标进入(dom)  mouseup 鼠标按键抬起  resize 窗口大小改变  scroll 网页滚动 |

(2). 页面加载完成事件：2次

a. DOM中: 先DOMContentLoaded，再window.onload

b. jq中: 先$(function(){ ... })，再$(window).load(function(){ ... })

c. 今后，所有不依赖于css和图片的初始化页面的jquery代码（事件绑定，设置元素的初始内容、属性和样式等），都应该放在$(function(){ ... })中

(3). 事件模型: 3个阶段: 捕获、目标触发、冒泡

(4). 获得事件对象: 定义事件绑定时:

$元素.事件名(function(e){ //触发事件时e=event }

(5). 取消冒泡/停止蔓延: e.stopPropagation()

(6). 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，但是必须用.on()绑定

b. 不用e.target代替this，this指回了最初触发事件的子元素，又可以用了

c. 为.on()添加第二个选择器参数，.on()自动判断当前点击的元素是否想要的。无需自己写if了。

$父元素.on("事件名", "选择器", function(){

//this->指向最初触发事件的子元素，不再指父元素，this又能用了

//不用写if了

})

(7). 阻止元素自带的默认行为: e.preventDefault();

(8). 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

(9). 鼠标事件:

a. DOM中: mouseover mouseout 反复进出子元素也会反复冒泡触发父元素的进出事件

b. jq中:

1). mouseenter和mouseleave 代替mouseover和mouseout

2). 如果同时绑定mouseenter和mouseleave，就可只绑定一个hover

$元素.hover( //=mouseenter+mouseleave

function(){ ... }, //给mouseenter

function(){ ... } //给mouseleave

)

3). 特例: 万一两个函数可以修改为一样的，也可以只写一个function

$元素.hover( //=mouseenter+mouseleave

function(){ ... } //给mouseenter和mouseleave

)

(10). 模拟触发: $元素.trigger("事件名")

如果要触发的事件属于常用事件列表，可简写为 $元素.事件名()

(11). 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->day02 7. 添加 删除 替换 克隆

8. jquery 事件 事件委托

小程序->在线->JQUERY->day03 1. 页面加载完成后自动执行

2. 鼠标事件 模拟触发

3. jquery 动画

作业:

1. 复习今日小程序问题清单(小程序->在线->JQUERY->day02、03)，预习明日小程序问题清单(小程序->在线->jQuery->day04)

2. 看小程序视频预习jQuery类数组对象操作:

小程序->在线->JQUERY->day03 4. 类数组对象操作... ...

3. 看小程序视频预习为jQuery添加自定义函数:

小程序->在线->JQUERY->day04 14. 添加jQuery自定义函数...... ...

4. 看小程序视频预习用jquery发送ajax请求:

小程序->在线->JQUERY->day04 15. jQuery发送ajax请求

5.（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 ——最后一天

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