小程序: WEB问题速查

1. 问题清单: 小程序->首页->JQUERY->day03

2. 高清无码小视频: 小程序->在线->JQUERY->day03

今日扩展视频:

小程序->在线->JQUERY->day02

8. jquery 事件 事件委托 .... 优化标签页、删除方块...

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

不用今天都看完！！！！！！\*\*\*两个月后\*\*\*鄙视面试前\*\*\*看完就行\*\*\*！

3. 单词列表: 小程序->单词->JQUERY->day03

第三阶段上课约定:

1. 上课千万不要跟着记笔记和写代码！

2. 拼写错误！自己找！让我帮你找，你需要手抄20遍，给我拍照！

3. 现在正处在照猫画虎的阶段！每个DOM效果至少写3遍:

3.1 注释+代码抄一遍！

3.2 保留注释，删除代码，把代码再填回来！

3.3 删除注释和代码，试着用自己的话把注释写回来，根据自己的注释，把代码填回来

一个DOM效果，只有自己能写出人话注释，才算自己会！——标准！

如何提问:

1. 上课时，案例中，笔记中那句话哪行代码看不懂，立刻微信私信问我！

2. 小程序中问题清单里哪个问题和回答看不懂！立刻微信私信问我！

3. 个人项目中哪个功能不会做！立刻微信私信问我！

4. 刷笔试题时，遇到不会的笔试题！立刻微信私信问我！

下载源代码: 右键->链接另存为！

正课:

1. 事件

2. 动画

3. 类数组对象操作

4. 添加自定义函数

一. 事件:

1. 事件委托: (了解)

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

(1). 何时: 多个平级子元素都需要绑定相同的事件时

(2). 回顾DOM中: 3步

a. 事件只绑定在父元素上一份即可！

b. e.target代替this

c. 判断e.target是否想要的元素

(3). jq中: 3步:

a. 事件只绑定在父元素上一份即可！但是，必须用.on()方式绑定

b. this又重新指向了目标元素！不再需要e.target!

c. 判断目标元素是否想要的元素。但是，不需要我们自己判断。我们只要给.on()提供第二个选择器参数值，作为判断条件即可！.on()可自动用我们给的选择器条件，判断当前元素是否想要的！

$父元素.on("事件名", "选择器", function(){

~~e.target~~

this -> 目标元素

~~if(判断目标元素是否想要的){~~

~~}~~

})

(4). 问题: 这种简写只有jQuery能用！无论DOM，还是将来的各种框架！都不能用！

(5). 所以: DOM中标准的事件委托3步才是最重要的！将来框架如果使用事件委托优化，用的也是DOM中的标准事件委托3步！！

(6). 示例: 使用jquery事件委托简化DOM的事件委托三步

event/9\_append2.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>      //功能1: 点击按钮添加新方块      //DOM4步      //1. 查找触发事件的元素      //本例中: 查找按钮      // id选择器      $("#add-block")//因为只用一次，所以不用放变量      //2. 绑定事件处理函数      .click(function(){        //3. 查找要修改的元素        //4. 修改元素        //4.1 使用HTML片段创建一个新的块元素        $(`<div class="block">          <span class="close">×</span>        </div>`)//新块元素        //4.2 设置块元素的背景色为随机颜色        .css("background-color",`rgb(${          parseInt(Math.random()\*256)//每个颜色值从0~255        },${          parseInt(Math.random()\*256)//每个颜色值从0~255        },${          parseInt(Math.random()\*256)//每个颜色值从0~255        })`)        //4.3 将新创建的块插入到div的开头！        .prependTo(".container");//自动查找父元素！      });      //功能2: 点x，删除x所在的块元素      //DOM4步      //1. 查找触发事件的元素      //解决: 本来多个平级的x按钮都需要单击。事件应该只绑定在父元素上一份即可！所有x按钮都冒泡使用父元素上的单击事件处理哈函数！——复习DOM第四天！      $(".container")      //2. 绑定事件处理函数      //           类选择器: 只希望class为close的按钮才能触发事件      .on("click",".close",function(){        //this->e.target        //3. 查找要修改的元素        //4. 修改元素        //删除当前×按钮的父元素块        $(this).parent().remove();      })    </script>     </body>  </html>  运行结果: |

2. 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: 如果希望在页面加载后就自动执行一段代码(事件绑定、发送ajax请求)，这段代码应该写在哪里？

(2). 解决: 习惯上放在<body>结尾的<script>中

(3). 问题2: 将js代码放在.html文件中，不符合内容与行为分离的原则，不便于维护和重用！

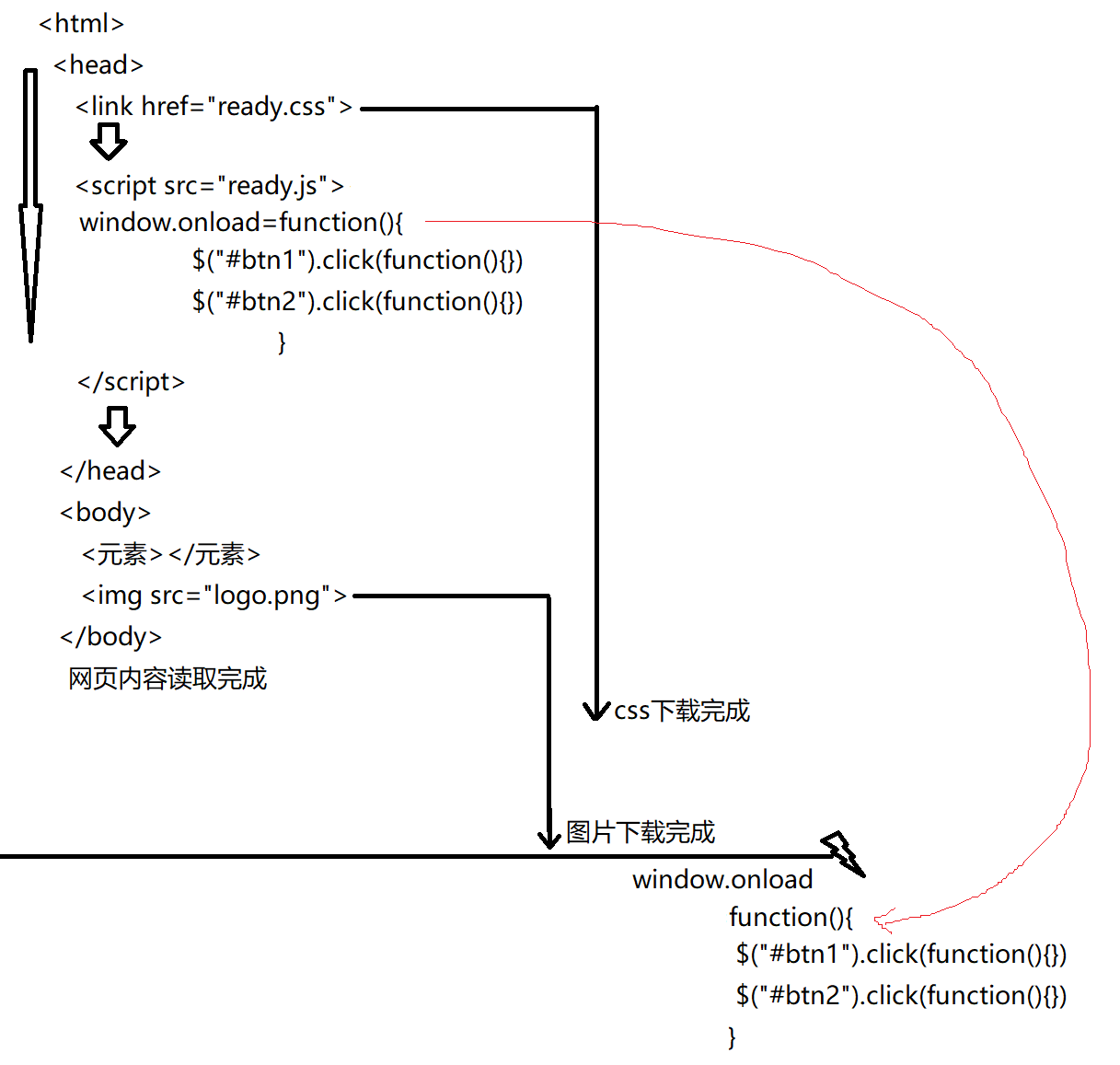
(4). 解决: 今后所有的js代码都要放在专门的.js文件中保存，然后用<script>引入页面

(5). 问题3: 将来别人引入.js文件时不一定都放在body结尾！有的人就习惯在<head>中引入.js文件！但是，如果在body之前引入.js文件，执行其中的js代码，因为此时还没有解析body的内容，所以什么元素都找不到！什么操作都无法执行！

(6). 解决: 希望将来无论在何处引入.js文件，其中的js代码，注定都在页面加载完成后才自动执行！

a. 整个网页加载完成时，会自动触发浏览器上一个事件: window.onload

b. 如果放入window.onload事件处理函数中的代码，注定只能在整个页面内容(HTML+JS+CSS+图片)都加载完之后才自动执行。



(7). 问题4: 如果一个页面中引入了多个.js文件，而且多个.js文件中都包含window.onload=function(){ ... }，只有最后引入的.js文件中的window.onload事件处理函数得以保留！之前都被覆盖了

(8). 解决: 今后即便用window.onload，也要以添加事件监听对象的方式来避免覆盖

window.addEventListener("load",function(){ ... })

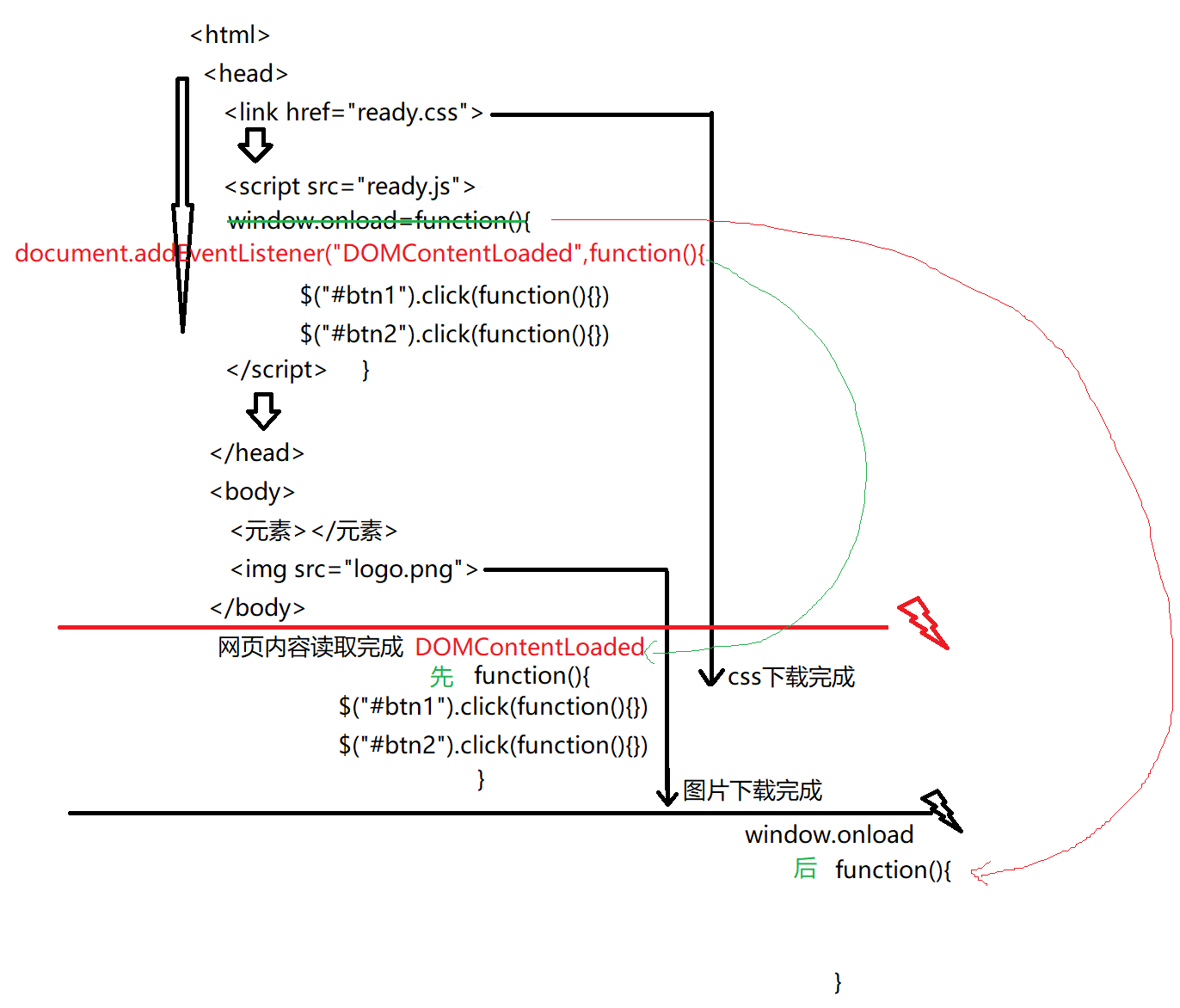
(9). 问题5: window.onload注定必须等待所有网页内容(HTML+JS+CSS+图片)都加载完才能执行！但是！有些非常着急的操作，是没必要等待样式和图片加载完，就要提前赶紧操作！（比如: 过年抢票！）

(10). 解决: 其实一个页面在加载过程中，在window.onload之前，还有一次加载完成事件: DOMContentLoaded（仅DOM内容加载完成就提前触发！）。这个事件只需要等待HTML+JS完成就可提前触发。无需等待CSS和图片！

document.addEventListener("DOMContentLoaded",function(){ ... })

(仅)DOM 内容 加载完成

(11). 结论: 今后凡是和css和图片无关的网页内容初始化js代码，都要放在DOMContentLoaded事件中。比如：事件绑定, 发送ajax请求... ...



(12). 问题6: 太长了！jquery一定提供了简写!

(13). 解决: jq中:

a. window.addEventListener("load",function(){ ... })可简写为

$(window).load(function(){ ... })

b. document.addEventListener("DOMContentLoaded",function(){ ... })可简写为:

1). 标准:$(document).ready(function(){ ... })

当网页内容准备就绪后自动执行

2). 简写:

i. 去掉document： $().ready(function(){ ... })

ii. 再去掉ready: $(function(){ ... })

iii. 最后改为箭头函数: $(()=>{ ... })

(14). 结论: 今后jq中，只要和css和图片无关的页面初始化操作的js代码，都要放在$(function(){ ... })里。如果浏览器支持ES6，可进一步简化为$(()=>{ ... })

(15). 强调: $(function(){ ... }) vs (function(){ ... })()

a. (function(){ ... })(): 匿名函数自调，会立刻执行！

b. $(function(){ ... }): 不是调用函数的代码，而是事件绑定的代码。不会立刻执行！只在DOMContentLoaded事件发生后，才自动执行！

(16). 示例: 在DOM内容加载完成后就为提前为按钮绑定单击事件:

event/14\_ready.js

|  |
| --- |
| //仅DOM内容(HTML+JS)加载完成，就提前  // document.addEventListener("DOMContentLoaded",function(){  //$(document).ready(function(){  // $().ready(function(){  // $(function(){  $(()=>{    alert(`DOMContentLoaded触发！`)    //为第一个按钮绑定单击事件    $("#btn1").click(function(){      console.log(`btn1 疼!`)    });    //用户就可以提前用上！  })  //整个页面内容都加载完成后自动执行以下操作  //window.onload=function(){  //window.addEventListener("load",function(){  $(window).load(function(){    alert(`window.onload触发！`)  }) |

14\_ready2.js

|  |
| --- |
| //仅DOM内容(HTML+JS)加载完成，就提前  // document.addEventListener("DOMContentLoaded",function(){  $(()=>{    //为第2个按钮绑定单击事件    $("#btn2").click(function(){      console.log(`btn2 疼!`)    });    //用户就可以提前用上！  })  //整个页面内容都加载完成后自动执行以下操作  //window.onload=function(){  window.addEventListener("load",function(){  }) |

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>      //在onload事件中显示整个页面加载完成的时间      //在ready事件中显示DOM内容加载完成的时间    </script>    <script src='14\_ready.js'>      //引入后立即执行！      //window.addEventListener("load",function(){        //只能在整个页面加载完成后才能执行！        //$("#btn1").click(function(){ ... })      //}    </script>    <script src="14\_ready2.js">      //window.addEventListener("load",function(){        //只能在整个页面加载完成后才能执行！        //$("#btn2").click(function(){ ... })      //}    </script>  </head>  <body>    <button id="btn1">click me 1</button>    <button id="btn2">click me 2</button>    </body>  </html>  运行结果: |

3. 鼠标事件:

对应小程序视频: 小程序->在线->JQUERY->day03 2. 鼠标事件

(1). 问题: DOM中的鼠标进入（mouseover）和鼠标移出(mouseout)两个事件有个小问题: 反复进出子元素，也会反复触发父元素上的鼠标进入和鼠标移出事件，与现实不符，会产生歧义

(2). 解决: jq中以及今后的框架中，都是用mouseenter（鼠标进入）和mouseleave(鼠标离开)代替mouseover和mouseout

(3). 优点: 反复进出子元素也不会反复触发父元素上的事件处理函数。与实际相符，不会产生歧义。

(4). 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>      //只在最外层d1绑定鼠标进入和鼠标移出事件      $("#d1")      //.mouseover(function(){      .mouseenter(function(){        console.log(`进入d1`)      })      //.mouseout(function(e){      .mouseleave(function(){        console.log(`移出d1`)      })    </script>  </body>  </html>  运行结果: |

(5). 问题: mouseenter和mouseleave经常成对儿出现！但是每次都写mouseenter和mouseleave时，太长了！

(6). 简写: jq中绑定一个.hover()事件，等效于同时绑定了mouseenter和mouseleave两个事件。但是，也必须同时给.hover()两个处理函数。其中第一个处理函数默认给mouseenter，第二个处理函数默认给mouseleave

$元素.hover( = mouseenter+mouseleave

function(){ ... }, //给mouseenter

function(){ ... } //给mouseleave

)

(7). 特殊情况下的更简写: 如果刚巧鼠标进入的事件处理函数和鼠标移出的事件处理函数可以改为完全一样的事件处理函数！则只需写一个处理函数，就可同时分给鼠标进入事件和鼠标移出事件！

$元素.hover( =mouseenter+mouseleave

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

)

(8). 示例: 使用hover简写mouseenter和mouseleave

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;      }      #target.hover {        border: 1px solid #aaa;        box-shadow: 0 0 6px #aaa;        color:#fff;        background-color:red;      }    </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>      //当鼠标进入id为target的div时，div添加hover class高亮显示      //当鼠标移出id为target的div时，div移除hover class恢复原样      $("#target")      // .mouseenter(function(){      //  $(this).addClass("hover")      // })      // .mouseleave(function(){      //  $(this).removeClass("hover")      // })      // .hover(      //  function(){      //    $(this).addClass("hover")      //  },      //  function(){      //    $(this).removeClass("hover")      //  }      // )      .hover(        function(){ $(this).toggleClass("hover") }      )    </script>   </body>  </html>  运行结果: |

4. 模拟触发:

对应小程序视频: 小程序->在线->JQUERY->day03 3. 模拟触发

(1). 什么是: 即使没有点在元素上，也能触发元素的单击事件

(2). 如何:

a. 标准: $元素.trigger("事件名")

触发

b. 简写: 如果要触发的事件名在常用的16种事件之中，才可简写为:

$元素.事件名()

(3). 示例: 按回车和点百度一下都能搜索

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>      //点击按钮获得文本框中用户输入的关键词      $("button").click(function(){                //复习jquery第一天        var kw=$(":text").val();//复习jquery第二天        //    复习js高级第三天        if(kw.trim()!==""){          console.log(`搜索 ${kw} 相关的信息...`)        }      });      //也想实现在文本框上按回车键时，执行和点击按钮一模一样的查找操作      //复习jquery第一天      //         键盘按下      $(":text").keydown(function(e){        //e.keyCode专门接受这次按的按键号的特殊属性        //console.log(e.keyCode)        //只有按的是回车键时，才执行搜索操作        if(e.keyCode==13){          //找到搜索按钮，并触发按钮上的单击事件处理函数          //$("button").trigger("click");          $("button").click();        }      })    </script>  </body>  </html>  运行结果: |

二. 动画:

对应小程序视频: 小程序->在线->JQUERY->day03 4. jquery动画

1. 简单动画: 3种写死的固定的动画效果：

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

显示 隐藏 在显示/隐藏之间来回切换

a. 强调: 显示隐藏相关的三个动画函数，如果不提供任何实参值，默认采用的是display:none/block方式显示隐藏，是没有过渡效果的！

b. 解决: 要想有过渡效果，就必须为这三个动画函数提供"动画持续时间"毫秒数参数值！

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

// 幕布 上滑 下滑 在上滑/下滑之间来回切换

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

// 淡入 淡出 在淡入/淡出之间来回切换

(4). jQuery简单动画的两大致命缺点:

a. 底层是用DOM操作+js定时器模拟的过渡效果！——效率远不如css的transition！

b. 都是在公共的动画函数中写死的动画变化过程！——几乎不可维护！

(5). 当真正的专业的前端工程师出现后，这一组简单动画函数，用的越来越少了！今后只要页面上简单的动画，都首选CSS的transition和animation实现！

(6). 特例: .show() .hide() .toggle()三个函数在不加毫秒数参数时，我们还是强烈推荐使用的！因为他们用的是display:none来显示隐藏元素，没有效率和可维护性的损失。

(7). 示例：1\_show\_hide.html

|  |
| --- |
| <!DOCTYPE html>  <html>   <head>    <title> new document </title>    <meta charset="utf-8">    <style>      \*{margin:0; padding:0;}      #target{        border-radius:10px;        background:#eee;      }      .fade{/\*动画起始状态\*/        height:104px; width:970px; opacity:1;        padding: 10px; overflow:hidden;        border: 1px solid #aaa;      }      .out{/\*动画结束状态\*/        }    </style>   </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，就让div显示出来        // if($div.css("display")=="none"){        //  $div.show(2000);        // }else{//否则如果div是显示的，就让div隐藏起来        //  $div.hide(2000);        // }        //$div.toggle();        $div.toggle(2000);        //$div.slideToggle()        //$div.fadeToggle()      });    </script>   </body>  </html>  运行结果: |

(8). 示例: 显示隐藏品牌列表:

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    $("button")    //2. 绑定事件处理函数    .click(function(){      //3. 查找要修改的元素      var $others=$("ul>li:gt(4):not(:last-child)");      console.log($others);      //4. 修改元素      //如果他们现在是隐藏的，就显示出来      if($others.css("display")=="none"){        $others.show();        //如果其他li都显示出来,就修改按钮的内容为"精简显示品牌"        $(this).html(`精简显示品牌`);      }else{//否则如果他们现在是显示的，就隐藏起来        $others.hide();        //如果其他li都隐藏起来，就修改按钮的内容为"显示所有品牌"        $(this).html(`显示所有品牌`);      }      //$others.toggle();    })  </script>  </body>  </html>  运行结果: |

2. 万能动画: 自由的对多种css执行动画效果！

(1). 如何:

$元素.animate({

要变化的css属性: 目标值,

... : ... ,

}, 动画持续时间ms数);

(2). 原理: animate()会自动当前元素要变化的属性的当前状态值！然后，用js程序计算出变化到目标值所经历的过程！然后，在规定的动画持续时间段内，执行动画效果！

(3). 说明:

a. animate()只支持属性值为单个数值的css属性:

width/height top/left/right/bottom margin/padding opacity font-size...

b. animate()不支持:

1). 不支持非数值的属性: font-family ...

2). 不支持颜色动画

3). 不支持CSS3变换: transform: rotate(90deg)

c. vs css transition: 既支持颜色过渡，又支持CSS3变换！

(4). 结论: 如果一个动画效果既能用css做，又能用js做，则一定首选css做！

(5). 底层: animate()也是用DOM+js定时器模拟的动画效果！——效率也不如css

(6). 示例: 测试animate()支持哪些css属性，不支持哪些css属性

4\_animate.html

|  |
| --- |
| <!DOCTYPE html>  <html>   <head>    <title> new document </title>    <meta charset="utf-8">    <style>      #d1{        border:1px solid #aaa;        border-radius:6px;        background:#eee;        width:50px; height:50px;        position:absolute; top:120px; left:0;      }    </style>   </head>   <body>    <h1>animate</h1>    <button id="btn1">启动动画</button>    <div id="d1">abcd</div>    <script src="js/jquery-1.11.3.js"></script>    <script>      $("#btn1").click(function(){        $("#d1").animate({          //width:400,//height          // left:400,//top right bottom          // opacity:0          // fontSize:64,          padding:50,//margin          backgroundColor:"red",          transform:"rotate(90deg)"        },2000)      })    </script>   </body>  </html>  运行结果 |

3. 排队和并发:

(1). 并发: 多个css属性同时变化

放在一个animate()内的多个css属性默认是并发变化的

(2). 排队: 多个css属性先后依次变化

a. 对一个元素先后调用多个animate()函数，多个animate()函数中的css属性是排队变化的！

$元素.animate({...}).animate({...})...

b. 原理:

1). 每个jq对象都有一个动画队列

2). 对jq对象调用动画函数(简单动画和animate())，其实不是立刻执行动画的意思！而是将动画临时添加到当前js对象的队列中等待！

i. 如果当前jq对象的动画队列中之前已经有正在播放的动画，则后续动画必须等待前一个动画执行完，才能继续执行

ii. 如果当前jq对象的动画队列中之前没有其他正在播放的动画，则新加入队列的动画才能立刻执行！

4. 停止动画:

(1). $元素.stop()

(2). 问题: .stop()默认只能停止一个正在播放的动画。如果队列中包含多个动画，则停止前一个动画，后一个动画依然继续播放！

(3). 解决: .stop(true) //停止当前动画，并清空动画队列！

5. 判断一个元素是否正在播放动画:

(1). jq中新增了一个选择器":animated"，专门匹配正在播放animate()动画的元素！

(2). jq中新增了一个函数: var bool=$元素.is("选择器")，专门判断当前元素是否符合选择器的要求！

(3). 所以，判断一个元素是否正在播放动画: var bool=$元素.is(":animated")

a. 如果返回true，说明正在播放动画

b. 如果返回false，说明没有在播放动画

6. 动画播放完成后自动执行:

(1). 错误做法: 将后续操作仅仅是放在动画函数之后

(2). 原因: 动画函数底层都是js定时器——异步的！仅仅放在动画函数之后的代码，不会等待动画函数执行完，就已经提前执行了

(3). 正确做法: jq的动画函数，都提供了最后一个回调函数参数！

凡是写在动画函数最后一个回调函数参数中的代码，注定只能在动画函数结束后才能自动执行

(4). 如何: $元素.动画函数({...}, 动画持续时间, function(){ ... })

7. 示例: 四颗小星星，点每个星星，执行规定的不同动画

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 $this=$(this);//因为要反复使用了！        //如果当前星星正在播放动画，就停止动画        if($this.is(":animated")){          $this.stop();        }else{//否则如果当前星星没有在播放动画，就启动动画          $this.animate({            left:400          },2000)        }      })      $("#s2").click(function () {        var $this=$(this);//因为要反复使用了！        //如果当前星星正在播放动画，就停止动画        if($this.is(":animated")){          $this.stop(true);        }else{          //动画队列=========================================          $this.animate({left:400},2000).animate({top:200},1000)          //        播放                      等待          //================================================        }      })      $("#s3").click(function(){        var $this=$(this);//因为要反复使用了！        //如果当前星星正在播放动画，就停止动画        if($this.is(":animated")){          $this.stop();        }else{          $this.animate({            left:400,            top:200          },3000)        }      })      $("#s4").click(function(){        alert("疼!")        $(this).animate({//animate底层是js定时器——异步          width:256,          height:256,          opacity:0        },2000,function(){          //要求动画结束后，将第四颗星星，彻底隐藏          $(this).hide();//display:none        })        //直接放在animate()之后的代码，不会等待animate()执行完，就提前执行了！      })    </script>  </body>  </html>  运行结果: |

三. 类数组对象操作:

1. 问题: jQuery的查找结果对象，是一个类数组对象！但是，类数组对象在js中，很受歧视！类数组对象长的像极了数组，但是无法使用数组家任何函数！如果jQuery对象也无法使用任何数组家的函数，则会极大的限制jquery的功能！

2. 解决: jquery仿着数组家的两个最常用的函数，为jquery对象也添加了两个功能完全一样的简化版函数: 遍历each()和查找index()

3. 遍历: 仿着数组家的forEach做出了each()

(1). 什么是: 依次取出jQuery查找结果对象中每个DOM元素对象，执行相同的操作！

(2). 如何:

a. 回顾: 数组forEach()

//复习JS高级第6天

// 当前元素值, 当前位置, 当前数组对象

arr.forEach(function(elem, i , arr){

//没遍历到一个元素，就自动调用一次

})

b. jq中: 当前位置, 当前DOM元素对象

$(...).each(function( i, domElem){

//每遍历到一个DOM元素对象，就自动调用一次回调函数

})

(3). 示例: 遍历ul下每个li，执行相同的操作

9\_each.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">    <meta charset="UTF-8">    <title></title>    <style>    </style>  </head>  <body>  <ul id="list">    <li>98</li>    <li>85</li>    <li>33</li>    <li>99</li>    <li>52</li>  </ul>  <script src="js/jquery-1.11.3.js"></script>  <script>    var arr=["亮亮","然然","东东"];    arr.forEach(function(elem){      console.log(`${elem} - 到！`)    })    //请给每个不足60分的成绩+10分，并将超过90分的成绩用绿色背景标识出来    $("ul>li").each(function(i,elem){      //先包装elem为$elem      var $elem=$(elem);      //获得当前li中的分数，转为整数      var score=parseInt($elem.html());      //如果当前li的内容<60，就加10分      if(score<60){        score+=10;        $elem.html(score);//再放回去      }else if(score>=90){//否则如果当前li的内容>=90，就设置当前li的背景为绿色！        $elem.css("background-color","green");      }    })  </script>  </body>  </html>  运行结果: |

4. 查找: 仿着数组家indexOf()函数实现了index()

(1). 什么是: 查找一个元素，在整个查找结果集合中的下标位置

(2). 如何:

a. 回顾数组家indexOf()

var i=arr.indexOf(元素值)

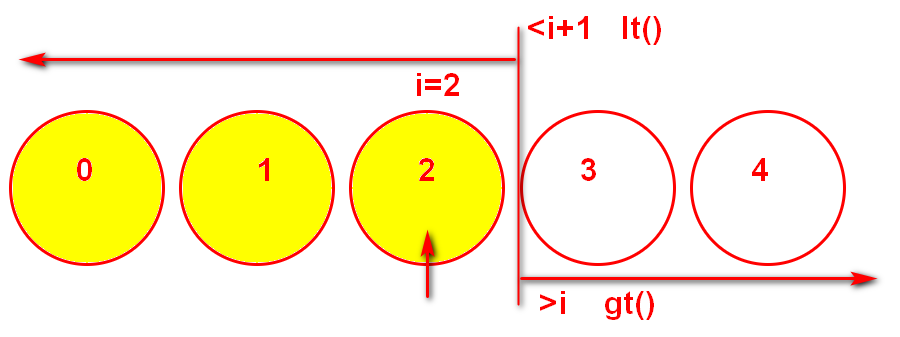
查找指定的元素值在arr数组中的下标位置！

b. jq中index()

var i=$(...).indexOf(DOM元素)

查找一个指定的DOM元素在整个查找结果集合中的下标位置

(3). 示例: 5星评价:



10\_index.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">    <meta charset="UTF-8">    <title></title>    <style>      .score {        list-style: none;        margin: 0;        padding: 0;      }      .score li {        display: inline-block;        width: 50px;        height: 50px;        border: 1px solid #f00;        border-radius: 50%;        cursor: pointer;      }    </style>  </head>  <body>  <h3>请打分</h3>  <ul class="score">    <li></li>    <li></li>    <li></li>    <li></li>    <li></li>  </ul>  <script src="js/jquery-1.11.3.js"></script>  <script>    var arr=[1,2,3,4,5];    //       0 1 2 3 4    //想知道3在数组中的哪个下标位置    var i=arr.indexOf(3);    console.log(`3在数组中下标为[${i}]的位置`);    //获得当前单击的li在所有li中的位置i,i及其执行的都变为红色，i之后的都变为白色    //DOM4步    //1. 查找触发事件的元素    //本例中ul下所有里都能点击    var $lis=$("ul>li");    //2. 绑定事件处理函数    $lis.click(function(){      //3. 查找要修改的元素      //4. 修改元素      //先获得当前点击的时第几个li      //其实就是在所有li中找当前li的下标位置      var i=$lis.index(this);      console.log(i);      //让<i+1位置的所有li背景变为黄色      //           变量      $(`ul>li:lt(${i+1})`)        .css("background-color","yellow");      //让>i位置的所有li背景变为白色      $(`ul>li:gt(${i})`)        .css("background-color","#fff")    })  </script>  </body>  </html>  运行结果: |

总结:

|  |
| --- |
| jQuery中$的原理: 4种:  1. $("选择器") 查找元素，并创建jQuery对象  2. $(DOM元素) 不用查找，直接将一个DOM元素包装为jQuery对象  3. $(`HTML片段`) 创建新元素，并保存进jQuery对象中  4. $(function(){ ... }) 绑定DOMContentLoaded事件处理函数！ |

|  |
| --- |
| jQuery简化版函数3大特点:  (1). 自带for循环: 对整个jquery子对象调用一次简化版函数，等效于对jQuery子对象中保存的每个DOM元素对象分别调用一次对等的DOM原生方法或属性——不用自己写for循环  (2). 一个函数两用: 调用函数时:  a. 没给新值作为参数，默认执行获取旧值的操作  b. 给了新值作为参数，自动切换为执行修改操作  (3). 本没有返回值的函数，默认返回当前正在操作的.前的jQuery对象——可链式操作 |

总结:

(3).用节点间关系查找: 2大类关系, 8个函数

a. 父子关系：3个函数:

$元素.parent()

$元素.children("选择器")

$元素.find("选择器")

b. 兄弟关系：5个

$元素.prev()

$元素.prevAll("选择器")

$元素.next()

$元素.nextAll("选择器")

$元素.siblings("选择器")

6. 修改元素: 3种:

修改中的所有函数，都是一个函数两用！

(1). 内容: 3种:

a. 原始HTML内容:

$元素.html("新HTML内容") 代替.innerHTML

b. 纯文本内容:

$元素.text("纯文本内容") 代替.textContent

c. 表单元素的值:

$元素.val("新值") 代替.value

(2). 属性: 3种:

a. 字符串类型的HTML标准属性:2种:

1). $元素.attr("属性名","新属性值")

代替 元素.getAttribute()和setAttribute()

2). $元素.prop("属性名", bool值)

代替 元素.属性名=bool值  
b. bool类型的HTML标准属性: 只有1种

$元素.prop("属性名", bool值)

代替 元素.属性名=bool值

c. 自定义扩展属性:只有一种:

$元素.attr("属性名","新属性值")

代替 元素.getAttribute()和setAttribute()

反过来总结:

$元素.attr()可修改一切字符串类型的属性(字符串类型HTML标准属性+自定义扩展属性)

$元素.prop()可修改一切可用.访问的属性(所有HTML标准属性)

(3). 样式:

a. 获取或修改单个css属性: 只有1种

$元素.css("css属性名", "属性值")

获取属性值时被翻译为getComputedStyle(元素)

修改属性值时被翻译为.style.css属性=属性值

b. 使用class批量修改样式:

$元素.addClass("class名")

$元素.removeClass("class名")

$元素.hasClass("class名")

$元素.toggleClass("class名")

|  |
| --- |
| 修改相关的函数都可同时修改多个属性值:  $元素.attr或prop或css({  属性名:"属性值",  ... : ...  }) |

8. 添加删除替换克隆元素:

(1). 添加新元素: 2步

a. 使用HTML片段批量创建新元素:

$(`HTML片段`)

b. 将新元素添加到DOM树: 5种方式，10个函数

1). 末尾追加:

$父元素.append($新元素)

$新元素.appendTo($父元素)  
 2). 开头插入: 新增:   
 $父元素.prepend($新元素)

$新元素.prependTo($父元素)

3). 插入到一个现有元素之前:

$现有元素.before($新元素)

$新元素.insertBefore($现有元素)

4). 插入到一个现有元素之后:

$现有元素.after($新元素)

$新元素.insertAfter($现有元素)

5). 替换现有元素:

$现有元素.replaceWith($新元素)

$新元素.replaceAll($现有元素)

(2). 删除元素: $元素.remove()

(3). 克隆元素: $元素.clone()

9. 事件绑定:

(1). 标准写法: $元素.on("事件名", 事件处理函数)

(2). 简写: $元素.事件名(事件处理函数)

(3). 事件委托:

$父元素.on("事件名","选择器",function(){

...this指向e.target...

})

(4). 页面加载后自动执行:

a. 先$(document).ready(function(){})

简写: $(function(){ ... })

b. 后$(window).load(function(){ ... })

(5). 鼠标事件:

mouseenter代替mouseover

mouseleave 代替mouseout

简写: .hover(处理函数1,处理函数2)

等于: .mouseenter(处理函数1)

.mouseleave(处理函数2)

(6). 模拟触发:

a. 标准: $元素.trigger("事件名")

b. 如果属于常用事件列表，可简写为:

$元素.事件名()

10. 动画:

(1). 简单动画: 3种固定效果

a. 显示隐藏:

.show() .hide() . toggle()

b. 淡入淡出:

.fadeIn() .fadeOut() .fadeToggle()

c. 上滑下滑:

.slideUp() .slideDown() .slideToggle()

(2). 万能动画:

$元素.animate({

css属性: 目标值

},动画持续时间,function(){

动画播放结束自动执行

})

(3). 排队和并发:

a. 放在一个animate()中的多个css属性并发变化

b. 放在一个元素的多个animate()中的多个css属性排队变化

(4). 停止动画:

a. 只停止当前一个动画

$元素.stop()

b. 停止队列中所有动画

$元素.stop(true)

(5). 选择器匹配正在播放动画的元素: :animated

11. 类数组对象操作:

(1). 遍历查找结果中每个DOM元素对象:

$查找结果.each(function(i, domElem){ ... })

(2). 查找一个DOM元素在整个查找结果中的下标位置

var i=$查找结果.index(要找的DOM元素)

今日对应小程序视频列表:

小程序->在线->JQUERY->day02 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

3. 模拟触发

4. jquery动画

5. 类数组对象操作

作业:

1. 复习今日小程序问题清单: 小程序->首页->JQUERY->day03

2. 看小程序视频和源代码实现规定功能:

小程序->在线->JQUERY->day03 3.1 2个作业: 精简显示品牌列表+星星闪烁

就在day03/ 2\_show\_hide2.html 和 6\_star.html中直接做即可

3. 看小程序视频预习: 小程序->在线->JQUERY->day04:

14. 添加jquery自定义函数

15. jquery发送ajax请求

16. jquery跨域

4.（凡是觉得知识点串不起来）看小程序视频 完成 jQuery版学子商城项目

小程序->在线->JQUERY->day04 项目: 0~4 学子商城项目...

所有人下载项目:0. 准备中的public\_start文件夹，是项目的起始状态（空的），看着视频中项目:1，项目:2, 项目:3,...跟着做。

public.zip中是完成的代码，如果作业过程中出错，可以跟public.zip中代码对一下。

家里环境xampp和nodejs能用，且第一阶段学的好的同学:

从小程序中项目: 0. 准备...中下载xzserver.zip，用自己的服务器端代码

家里环境xampp和nodejs不能用，或者第一阶段学的不好的同学：

无需下载xzserver.zip

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