小程序: WEB问题速查

1. 问题清单: 历届师兄师姐在今天课程中集中问过的高频的重点的问题！

那些自己发现不了问题，又迫切想知道其它同学都问了什么问题，的同学

小程序->首页->JQUERY->day03

当老师在回答其它同学的问题时，你不要闲着！你可以打开问题清单浏览别人曾经问过的经典问题！

2. 高清无码小视频: 小程序->在线->JQUERY->day03

2部分: 1. 课上重点知识和案例的回放！2. 重点高频笔试面试题讲解扩展视频

万一家里或中心网络断了！暂时看不了直播了！

因为某些事儿请假了，错过了当天的直播！

课下复习时，觉得TMOOC视频太大，浏览不方便或TMOOC不能看了！

——打开小程序中"在线"，就可下载观看当前所讲内容和当天所讲全部内容的回放。

基础比较好的，确实学有余力的同学，想拿高薪的同学，在我答疑时，不要闲着:

要看今天扩展视频:

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

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. 凡是拼写错误，自己懒得找！我可以帮你找，但是找到后，错误的单词你要手抄20遍，并拍照！

出错: 先开f12，console，看错误提示！

3. 每个带注释的案例至少写三遍:

(1). 注释+代码抄一遍

(2). 保留注释，删除代码，自己试着把代码翻译回来！

(3). 删除注释和代码，自己试着用自己的话把注释写回来！

一个例子只有自己可以把人话的注释写回来！才算自己会！

问题: 一听就会一做就废！原因: 你自己无法用人话精确描述自己想干什么！

如何提问: （vip待遇的剩余时间6天）

1. 上课时，案例中，笔记中那句话哪行代码看不懂，立刻问！

2. 小程序中问题清单里哪个问题和回答看不懂！立刻问！

3. 个人项目中哪个功能不会做！立刻问！

4. 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. 事件

2. 动画

3. 类数组对象操作

4. 添加自定义函数

5. \*\*\*封装自定义插件

一. 事件:

1. 利用冒泡/事件委托:

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

(1). DOM中: 3步——反而更通用

a. **事件只绑定在父元素上一份即可**

b. **用e.target代替this**

c. **判断e.target的特征是否是我们想要的**

d. 示例: 使用jquery和DOM事件委托3步优化标签页案例

event/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 事件委托3步        //1. 查找触发事件的元素        //本例中: 因为多个子元素a都需要能单击，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即可        $("ul.tabs")        //2. 绑定事件处理函数        //本例中: 单击触发变化/换成鼠标移入变化        .click(function(e){ //mouseover(function(e){          //e.target代替this          var $target=$(e.target);          //只允许data-toggle=tab的元素触发事件          if($target.attr("data-toggle")=="tab"){            //3. 查找要修改的元素            //本例中: 既改自己，又改别人            //4. 修改元素            //先给自己的爹加active class            $target.parent()            //当前a  当前li                  .addClass("active") //本没有返回值            //默认返回.前的当前li的jquery对象            //再去掉自己的爹的所有兄弟的active class            //下一句刚好再次需要用到自己的爹            //于是就可以链式操作                  .siblings()            //    除当前li之外的其它所有兄弟li                  .removeClass("active")          }        })      </script>   </body>  </html>  运行结果: |

(2). jq中: 简化 —— 仅限于jQuery中使用

a. 事件只绑定在父元素上一份即可，但是必须用

**$元素.on("事件名", ...)**

b. **this又重新指向了e.target，this又可以使用了**！

c. 不用自己写if判断，而是为on指定第二个选择器参数，由on自动检测当前元素是否符合要求：

**$元素.on("事件名", "选择器", 事件处理函数)**

判断条件

1). 只有符合选择器要求的元素，才继续触发事件处理函数。

2). 如果当前元素不符合选择器要求，则不触发事件处理函数

d. 示例: 使用jquery简化版事件委托优化标签页案例

event/7\_tabs\_jq.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 事件委托3步        //1. 查找触发事件的元素        //本例中: 因为多个子元素a都需要能单击，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即可        $("ul.tabs")        //2. 绑定事件处理函数        //本例中: 单击触发变化/换成鼠标移入变化        .on("click","[data-toggle=tab]",function(){          //3. 查找要修改的元素          //本例中: 既改自己，又改别人          //4. 修改元素          //先给自己的爹加active class          $(this).parent()          //当前a  当前li                .addClass("active") //本没有返回值          //默认返回.前的当前li的jquery对象          //再去掉自己的爹的所有兄弟的active class          //下一句刚好再次需要用到自己的爹          //于是就可以链式操作                .siblings()          //    除当前li之外的其它所有兄弟li                .removeClass("active")        })      </script>   </body>  </html>  运行结果: 同上 |

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: js代码应该写在页面的什么位置才能保证正确执行？

答: 应该写在HTML内容之后，body的结尾

(2). 问题: 按理说根据内容与行为分离的原则，为了便于维护，HTML文件中就不应该包含js代码

解决: 所有js代码都应该放在HTML外的独立的js文件中保存，然后用<script>标签引入页面中使用

(3). 问题: 将来公司里什么人都有，有人愿意把js在head里引入，有人愿意把js在body结尾引入。如何保证无论js在开头还是在结尾引入都能正确执行？

解决: 利用页面加载完成事件

(4). 页面加载完成事件:

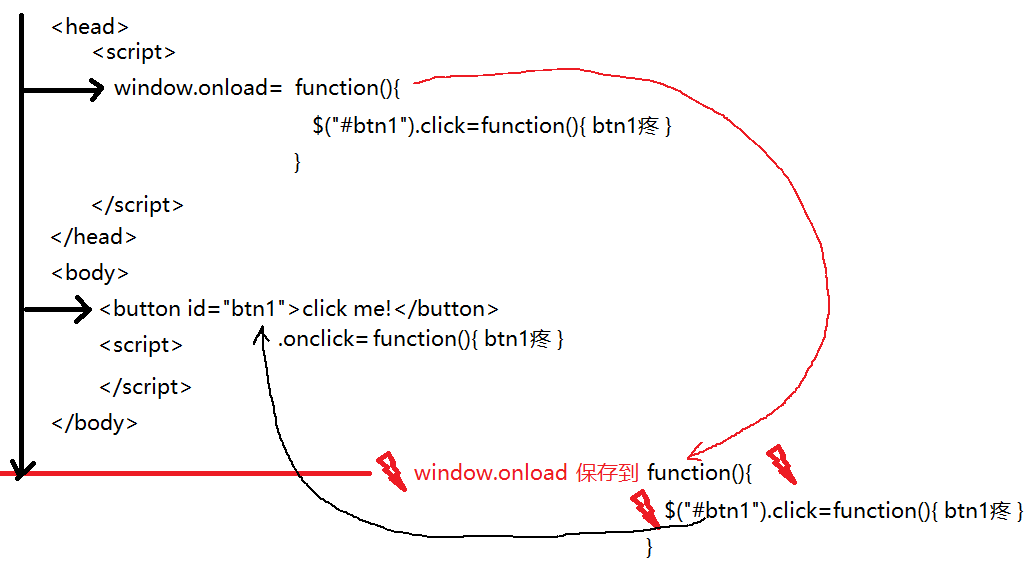
a. 其实在一个网页内容加载完成后，浏览器会触发一个事件。

b. 如果把我们的代码提前写在这个页面加载完成事件的处理函数中，提前保存在事件属性上。则无论任何情况下，我们的代码注定会随着事件**在整个页面加载完成后**才自动执行。

c. 如何: window.onload=function(){ ... 注定只会在整个页面加载完成后才自动执行...}

d. 强调: 只是事件绑定代码，不会立刻执行！只在事件发生时才自动执行！

(5). 总结: 将js代码放在**window.onload=function(){}**中，无论js代码在页面开头引用，还是在页面结尾引用，注定只会在整个页面加载完成后才自动执行，且都能正确执行

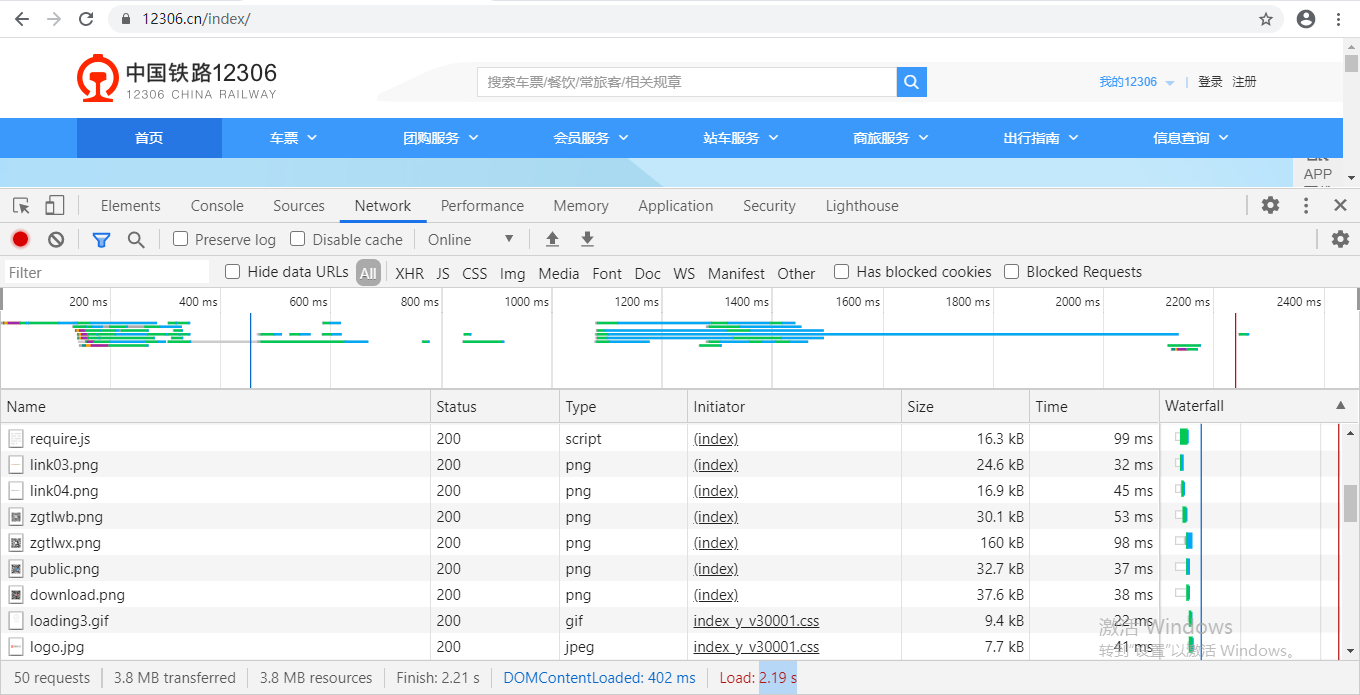


(6). 问题:不想让DOM和jQuery混搭

解决: 可将DOM的window.onload改为jq写法

window.onload=function(){} 改为 **$(window).load(function(){ ... })**

(7). 问题: window.onload必须等待所有页面内容（HTML+CSS+JS+图片）都加载完才能执行——有点儿晚！但是，有些情况下，用户着急使用功能，而不关心界面的好看难看！此时，就不应该强迫用户必须等待图片和css加载完才能使用网页！——比如: 春运抢票时12306



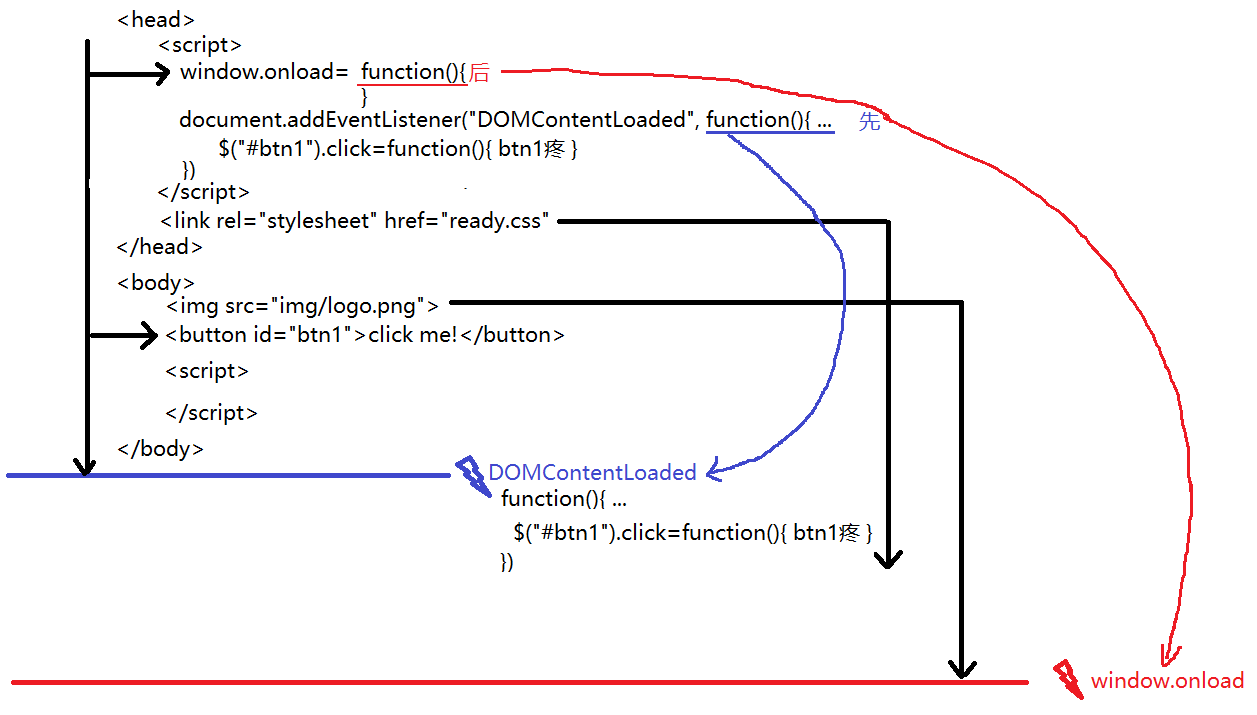
(8). 解决: 其实在window.onload之前，还会先触发一次另一个事件——仅**DOM内容加载后自动执行**。学名DOMContentLoaded。仅等待HTML+JS，无需等待CSS和图片。所以，DOMContentLoaded比window.onload要早触发！

(9). 如何绑定仅DOM内容加载完成就提前触发

a. DOM中: document.addEventListener("DOMContentLoaded",事件处理函数)

b. jq中: $(document).on("DOMContentLoaded",事件处理函数)

(10). 结论: 今后为了让用户尽早用上功能，而不用关心css和图片，大部分js代码都应该放在DOMContentLoaded中。



(11). 特例: 如果js代码中页面一加载就用到了css的样式或用到了图片，则只能放在window.onload中！

(12). 问题: 因为今后几乎所有的js代码都要放在DOMContentLoaded中，但是每次都写$(document).on("DOMContentLoaded",事件处理函数)，太麻烦！

(13). 解决: 简写:

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

已准备好

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

c. **$(function(){ ... })**

d. ES6: **$(()=>{ ... })**

(14). 大结局: 今后几乎所有jq代码都要放在$(function(){ ... })或$(()=>{ ... })中！

强调: 结尾一定不要画蛇添足加()！！！因为这是事件绑定，不是匿名函数自调！

(15). 示例: 演示window.onload和DOMContentLoaded事件的先后顺序

event/14\_ready.js

|  |
| --- |
| //仅绑定整个页面加载完成才自动执行的事件处理函数  //而不是立刻执行！  //window.onload=function(){ //DOM  $(window).load(function(){ //jq    alert("整个页面加载完成!");  });  //$(document).on("DOMContentLoaded",function(){  //$(function(){  $(()=>{    alert("仅DOM内容加载完成，就绑定事件！");    //事件绑定代码应该放在DOMContentLoaded中,才能早执行！    $("#btn1").click(function(){      alert("btn1 疼！")    })  }) |

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"></script>  </head>  <body>    <button id="btn1">click me!</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中的鼠标事件mouseover（鼠标进入）和mouseout（鼠标移出）两个事件有问题，反复进出子元素，同样会反复触发父元素上的鼠标进出事件——与现实不符！

(2). 解决: 今后jquery中以及其他框架中，都用:

a. **mouseenter**代替mouseover 表示鼠标进入

b. **mouseleave** 代替mouseout 表示鼠标离开

(3). 优点: 即使反复进出子元素，也不会反复触发父元素上的的进出事件处理函数——与现实相符！

(4). 简写: 如果**同时**绑定mouseenter和mouseleave，可简写为绑定hover

**$元素.hover(** //相当于.mouseenter(...).mouseleave(...)

**function(){ ... },**

**function(){ ... }**

**)**

(5). 示例: 使用mouseenter和mouseleave代替mouseover和mouseout，并用hover简写

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>      $("#d1")      //想只要鼠标进入外层d1的范围，就输出进入d1      //.mouseover(function(){      // .mouseenter(function(){      //  console.log(`进入d1`)      // })      //只要鼠标离开外层d1的范围，就输出离开d1      //.mouseout(function(){      // .mouseleave(function(){      //  console.log(`离开d1`)      // })      .hover(//.mouseenter().mouseleave()        function(){ console.log(`进入d1`) },        function(){ console.log(`离开d1`) }      );    </script>  </body>  </html>  运行结果: |

(6). **特殊**: 当两个事件处理函数刚好可以修改为完全相同的两个函数时，其实hover中也可以**只写一个函数**。但是这个函数既给mouseenter，又给mouseleave

$元素.hover( function(){ ... } )

.mouseenter(...) .mouseleave(...)

(7). 示例: 使用hover简写

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;      }      #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>      //希望鼠标进入id为target的div，让div高亮显示(添加自定义class hover)      //鼠标离开id为target的div，让div恢复正常显示（移除自定义class hover）      $("#target")      .hover(        function(){//进入时+.hover          //$(this).addClass("hover")  **$(this).toggleClass("hover")**        }//,        // function(){//离开时-.hover        //  //$(this).removeClass("hover");        //  $(this).toggleClass("hover")        // }      )    </script>   </body>  </html>  运行结果： |

4. 模拟触发:

(1). 什么是: 即使没有点到按钮，也能触发按钮的单击事件，执行相同的操作

(2). 比如: 百度：点按钮"百度一下"可以执行搜索，同时在文本框按回车，也可以执行相同的搜索操作。

(3). 如何:

a. 标准: **$元素.trigger("事件名")**

触发

b. 简写: 如果要触发的事件属于**常用**事件列表中

可简写为: **$元素.事件名()**

(4). 示例: 点按钮可以搜索，在文本框按回车也能搜索

event/17\_trigger.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">    <meta charset="UTF-8">    <title></title>    <style>    </style>  </head>  <body>    <input id="kw"><button>百度一下</button>    <script src="js/jquery-1.11.3.js"></script>    <script>      //点百度一下按钮，获取文本框中的内容并执行搜索操作      $("button").click(function(){        var kw=$("#kw").val().trim();                              //复习js高级第二天        if(kw!==""){          console.log(`查找 ${kw} 相关的内容...`)        }      })      //希望在文本框上按下回车键，执行按钮的单击事件操作      //   键盘按键按下——屌丝可能按住不放！      //$("#kw").keydown(function(e){      //   键盘按键抬起      $("#kw").**keyup**(function(e){        //只希望本次按的是回车键时才执行查找操作        //e.keyCode属性保存着本次按键键盘号        //只希望当按的是**回车键(13号)时**，才执行后续操作        if(e.keyCode==13){          //$("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 1. 页面加载完成后自动执行

1. 简单动画: 固定的写死的三种动画效果:

(1). 显示隐藏：.show() .hide() .toggle()

显示 隐藏 在显示和隐藏之间来回切换

a. 问题: 以上三个函数，不加参数时，默认使用display:none或block实现显示隐藏，所以是瞬间显示隐藏，没有动画效果！

b. 解决: 其实以上三个函数中可以指定动画持续时间参数，只要提供动画持续时间，则三个函数才带动画效果

(2). 淡入淡出: .fadeIn() .fadeOut() .fadeToggle()

淡入 淡出 在淡入淡出之间来回切换

(3). 上滑下滑: .slideUp() .slideDown() .slideToggle()

上滑 下滑 在上滑和下滑之间来回切换

|  |
| --- |
| 简单动画两个致命缺点:  1. 动画效果和变化过程都是在函数中写死的，整个项目共用一套简单动画函数——几乎不可维护！  2. 可以在f12中element中看到动画变化过程，说明这套函数使用js定时器和频繁DOM操作，模拟实现的动画效果——效率远不如css  结论: 以上简单动画函数，尽量少用！  今后，**凡是简单的过渡效果，都用css+transition实现！**2个好处:  1. 极其便于维护  2. 效率远比js定时器和DOM要高的多！  特例: .show() .hide() .toggle()这三个函数在没有实参时，相当于display:none和block，没有性能损失的。所以，不需要动画效果的瞬间显示隐藏，反而强烈建议用.show() .hide() .toggle()简写 |

(4). 示例: 测试简单动画各种效果:

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");        //想如果div是隐藏的        // if($div.css("display")=="none"){        //  $div.show(2000);//就让div显示        // }else{//否则如果div是显示的        //  $div.hide(2000);//就让div隐藏        // }        //$("#target").toggle(2000) //在显示和隐藏之间来回切换        //$("#target").fadeToggle(2000);        $("#target").slideToggle(2000)      });    </script>   </body>  </html>  运行结果: |

2. 万能动画: 能对多种css属性应用动画效果

(1). 如何:

**$元素.animate({**

**要变化的css属性: 希望变化到的目标值,**

**... : ...**

**}, 动画持续时间ms)**

(2). 原理: animate()自动获得元素当前的状态，然后根据我们指定的目标状态求出差值。然后在指定持续时间内，均匀改变元素的状态。

(3). 强调:

a. animate**只支持属性值为单个数值的css属性**.

比如: width height

opacity

top left right bottom

font-size

margin padding

... ...

b. 与transition相比:

1). animate()**不支持颜色动画**，transition支持

2). animate()**不支持CSS3变换**, transition 支持

(4). 优缺点:

a. 优点: 相对灵活，但是远不如css transition

b. 缺点: 底层用js定时器和频繁操作DOM树模拟动画——效率极低

(5). 示例: 测试animate()支持和不支持哪些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({          //测试支持哪些css属性          //width:300, //height          //opacity:0,          left:300, //top right bottom          //fontSize:32          //padding:60, //margin          //不支持哪些css属性          //backgroundColor:"red" //rgb(r,g,b)          //transform:`rotate(60deg)`//css中的函数调用语法        },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

在**一个animate()**中写的**多个css属性**，默认是并发变化

(2). **排队**: 多个css属性先后依次变化

对**同一个元素**先后调用**多个animate()**，先后调用的多个animate()中的css是排队变化

(3). 原理:

a. 其实每个元素对象都有一个自己的动画队列

b. 每次调用animate()时，不是立刻执行动画的意思，而是将动画加入到元素的动画队列中保存

c. 如果元素的动画队列中之前没有其它正在执行的动画，则新加入的动画才立刻执行

d. 如果元素的动画队列中还有未执行完的动画，则新加入的动画需要等待队列中之前的动画都执行完，才能执行。

4. 停止动画: **随时可停止动画**，是animate()相比transition的唯一优点

(1). **$元素.stop()**

(2). 问题: .stop()默认只能停止动画队列中**正在播放的一个动画**。队列中之后的动画，依然会触发播放

(3). 解决: **$元素.stop(true)**

停止当前并清空队列中剩余动画的意思

5. jq中独有选择器: **:animated** 匹配正在播放动画的元素

6. 动画结束后自动执行:

(1). 错误: 直接在$元素.aniamte()之后写下一步代码

(2). 因为animate()底层是js定时器，是异步的，之后的代码不会等待定时器执行完，就已经提前执行了

(3). 正确: 其实每个**动画函数都有最后一个参数**——**回调函数:**

凡是写在动画函数最后一个回调函数参数中的代码，注定只能在动画结束后才被自动调用执行！

(4). **$元素.animte(**

**{ css样式 },**

**动画持续时间ms，**

**function(){**

**//回调函数中的代码，注定只能在动画结束后才被自动调用执行**

**}**

**)**

7. 示例: 四颗小星星，点击后，执行不同的动画效果

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 $this=$(this);        //点一下启动，点一下停止，再点再启动...        //如果当前元素正在播放动画        if($this.is(":animated")){          $this.stop()//就停止        }else{//否则如果当前元素没有播放动画          //才启动动画          $this.animate({            left:400          },2000)        }      });      $("#s2").click(function(){        var $this=$(this);        //点一下启动，点一下停止，再点再启动...        //如果当前元素正在播放动画        if($this.is(":animated")){          $this.stop(true)//就停止        }else{//否则如果当前元素没有播放动画          //才启动动画          $this          .animate({            left:400          },2000)          .animate({            top:200          },1000)        }      });      $("#s3").click(function(){        var $this=$(this);        //点一下启动，点一下停止，再点再启动...        //如果当前元素正在播放动画        if($this.is(":animated")){          $this.stop()//就停止        }else{//否则如果当前元素没有播放动画          //才启动动画          $this          .animate({            left:400,            top:200          },3000)        }      })      $("#s4").click(function(){        alert("疼!");        $(this)        .animate(//底层js定时器模拟的动画效果——异步          {//参数1            width:256,            height:256,            opacity:0          },          2000,//参数2          function(){//参数3            //animate()回调函数中的this指animate().前的jquery对象。            $(this).hide();//不会等之前异步代码执行完才执行，而是立刻执行            //$(this).css("display","none");          }        );      })    </script>  </body>  </html>  运行结果: |

三. 类数组对象操作:

对应小程序视频: 小程序->在线->JQUERY->day03 4. 类数组对象操作

1. 问题: jQuery查找结果返回的都是类数组对象。但是js中类数组对象非常受歧视。因为类数组对象虽然长的像数组，但是不能使用数组家任何函数！如果这样，jquery的功能就会收到极大的制约。

2. 解决: jQuery为了便于自己类数组对象的操作，仿着js的数组家，定义了两个几乎一模一样的jquery函数

3. jq中遍历操作:

(1). js中数组家: forEach //依次取出数组中每个元素值，调用相同的回调函数，执行相同的处理。

遍历每一个 元素值 下标 当前数组对象

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

...

})

(2). jq中新增了一个遍历函数: each //依次取出jquery结果对象中保存的每个DOM元素对象，调用回调函数，对找到的每个DOM元素对象执行相同的操作

每一个 下标 DOM元素对象

**$查找结果.each(function(i, domElem){**

...

})

(3). 示例: 使用forEach点名，使用each遍历ul下每个li

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>    //回顾:forEach    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    $("#list>li").each(function(i, domElem){      //domElem是一个DOM家的元素对象，不是jq家孩子      //如果想用jq家简化版函数，得用$()包一下      var $domElem=$(domElem);      //如果当前元素的内容<60，就+10分      if($domElem.html()<60){        //先取出旧成绩转为整数        var score=parseInt($domElem.html())          //成绩  转换为整数         获取innerHTML内容        //成绩+10        score+=10;        //将新成绩放回当前元素内容中        $domElem.html(score);      }else if($domElem.html()>=90){        //否则如果当前元素的内容>=90，就背景变为绿色        $domElem.css("background-color","green")            //修改css属性 背景     色      绿色      }    })  </script>  </body>  </html>  运行结果: |

4. jq中查找操作:

(1). js中数组家: indexOf() 查找一个指定的元素值在数组中的下标位置

var i=arr.indexOf(要找的元素值)

(2). jq中新增了一个查找函数: index() 查找一个指定的DOM元素在查找结果中的**下标**位置

**var i=$查找结果.index(要找的DOM元素对象)**

(3). 示例: 评分

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>    //回顾:js中的index    var arr=["亮亮","然然","东东"];    //在数组arr中查找东东的下标位置    var i=arr.indexOf("东东")          //  位置 的    console.log(`东东在 ${i} 位置`)    //DOM 4步    //1. 查找触发事件的元素    //本例中: class为ul下的每个li都能点    $("ul>li")    //2. 绑定事件处理函数    .click(function(){      //获得当前点击的li      var $this=$(this);      //3. 查找要修改的元素      //4. 修改元素      //先将自己改为黄色      $this.css("background-color","yellow");      //  修改css属性 背景   颜色     黄色      //获得自己在所有li中的下标位置      // 在ul下的所有li中 找  当前li的位置      var i=$("ul>li").index($this);      //                位置      //alert(i);      //让<i位置的都变为黄色      $(`ul>li:lt(${i})`)            // less than            //   小  于        .css("background-color","yellow");      //让>i位置的都变为白色      $(`ul>li:gt(${i})`)            //greater than            //  大     于        .css("background-color","#fff")    })  </script>  </body>  </html>  运行结果: |

|  |
| --- |
| 补: 判断当前元素是否符合xxx条件  **var bool=$元素.is("任意选择器")**  **是 条件**  以为判断$元素是否符合"选择器"的要求！ |

jq中$共有4种用法:

1. $("选择器") 查找DOM元素，并保存进新创建的jquery对象中

2. $(DOM元素对象) 无需查找，直接将已经获得的DOM元素，保存进新创建的jquery对象中

3. $(`HTML片段`) 新创建一个HTML元素，保存进新创建的jquery对象中

4. $(function(){ ... }) 绑定DOMContentLoaded事件！

总结: jQuery API三大特点:

1. 自带遍历:

2. 一个函数两用:

3. 返回当前正在操作的.前的jQuery对象。——链式操作！

总结:

5. 查找元素:

(1). jQuery支持用所以CSS3选择器查找

(2). jQuery新增选择器:

基本过滤、内容过滤、可见性过滤:、表单元素过滤:   
(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:

1). $元素.parent()

2). $元素.children("选择器")

3). $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:

(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("data-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:

1). $元素.addClass("class名")

2). $元素.removeClass("class名")

3). $元素.hasClass("class名")

4). $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:

a. 添加新元素: 2步

1). 使用HTML片段批量创建新元素:

$(`HTML片段`)

2). 将新元素添加到DOM树: 5种方式，10个函数

i. 末尾追加:

$父元素.append($新元素)

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

$新元素.prependTo($父元素)

iii. 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)

iv. 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)

v. 替换现有元素:

$现有元素.replaceWith($新元素)

$新元素.replaceAll($现有元素)

b. 删除元素: $元素.remove()

c. 克隆元素: $元素.clone()

9. 事件绑定:

a. 标准写法: $元素.on("事件名", 事件处理函数)

b. 简写: $元素.事件名(事件处理函数)

c. 事件委托:

$父元素.on("事件名","选择器",function(){

...this指向e.target...

})

d. 页面加载后自动执行:

1). 先$(document).ready(function(){})

简写: $(function(){ ... })

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

e. 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)

f. 模拟触发:

1). 标准: $元素.trigger("事件名")

2). 如果属于常用事件列表，可简写为:

$元素.事件名()

10. 动画:

a. 简单动画: 3种固定效果

1). 显示隐藏:

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

2). 淡入淡出:

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

3). 上滑下滑:

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

b. 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

})

c. 排队和并发:

1). 放在一个animate()中的多个css属性并发变化

2). 放在一个元素的多个animate()中的多个css属性排队变化

d. 停止动画:

1). 只停止当前一个动画

$元素.stop()

2). 停止队列中所有动画

$元素.stop(true)

e. 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. jquery动画

4. 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