小程序: WEB问题速查

1. 问题清单: 小程序->首页->JQUERY->day04

2. 高清无码小视频: 小程序->在线->JQUERY->day04

今日扩展视频:

小程序->在线->JQUERY->day03

17. jquery 跨域 cors ... express脚手架

作业: 高频笔试题 jsonp 跨域

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->day04

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. 类数组对象操作

2. 添加自定义函数

3. 封装自定义插件

4. ajax

5. \*\*\*跨域

一. 类数组对象操作:

对应小程序视频: 小程序->在线->JQUERY->day03 5. 类数组对象操作

1. 问题: js中类数组对象非常受歧视，即使长得像数组，但是无法使用数组家任何函数。但是，jquery的所有查询结果对象，又都是类数组对象。如果无法访问数组家那么多好的函数，则会极大的限制jquery的功能发挥。

2. 解决: jquery为了自己查询结果的服务，仿着数组家最常用的两个函数，定义了两个功能一模一样的函数，供jquery子对象调用

3. 遍历:

(1). 数组家:

数组.forEach(function(当前元素值,当前位置,当前数组对象){

//可以对数组中每个元素执行相同的操作

})

(2). jq家:

$查询结果.each(function(当前位置, 当前元素值){

})

(3). 强调: 因为jquery查找结果对象中保存的是多个DOM元素对象。所以，用each遍历出来的每个元素值，都是一个DOM元素对象。——如果想对每个DOM元素对象继续使用简化版函数，必须$()包裹一下

(4). 示例: 使用jquery each函数遍历找到的每个DOM元素

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>      //复习JS高级day07      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,elem){      var $elem=$(elem);      if($elem.html()<60){        var score=parseInt($elem.html());        score+=10;        $elem.html(score);      }else if($elem.html()>=90){        $elem.css("background-color","green")      }    })    </script>  </body>  </html>  运行结果: |

4. 查找:

(1). 数组家:

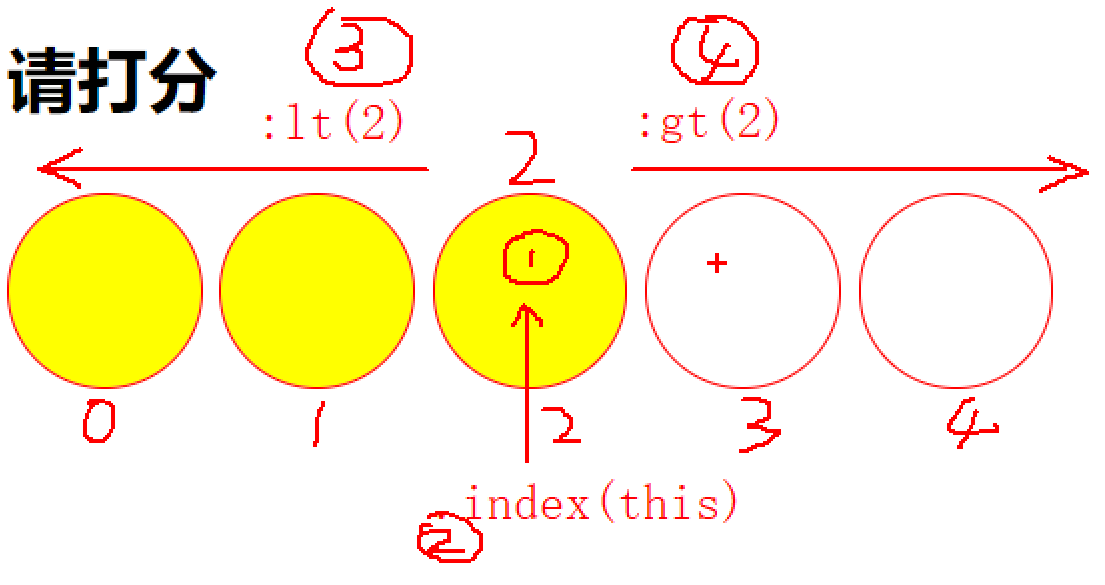
var i=数组.indexOf(元素值)

(2). jq家:

a. var i=$查找结果.index(要找的元素)

b. 在查找结果中查找一个指定过的DOM元素的下标位置。

(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    //查找元素3的下标位置是几    var i=arr.indexOf(3);    console.log(i);//2      //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    //DOM 4步    //1. 查找触发事件的元素    //本例中: 5个li都可单击    $("ul.score>li")    //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      var $this=$(this);      //1. 先让自己变黄色      $this.css("background-color","yellow");      //2. 获得当前li在所有li中的下标位置      var i=$("ul.score>li").index($this);      //3. <i位置的元素都变成黄色      //                 变量      $(`ul.score>li:lt(${i})`)      .css("background-color","yellow");      //4. >i位置的元素都变成白色      $(`ul.score>li:gt(${i})`)      .css("background-color","#fff");    })  </script>  </body>  </html>  运行结果: |

二. 添加自定义函数:

对应小程序视频: 小程序->在线->JQUERY->day04: 14 添加jQUery自定义函数...

1. 问题: 如果项目中经常需要对jquery查找结果对象执行一项操作，但是，jquery家却没有提供执行这项操作的函数。

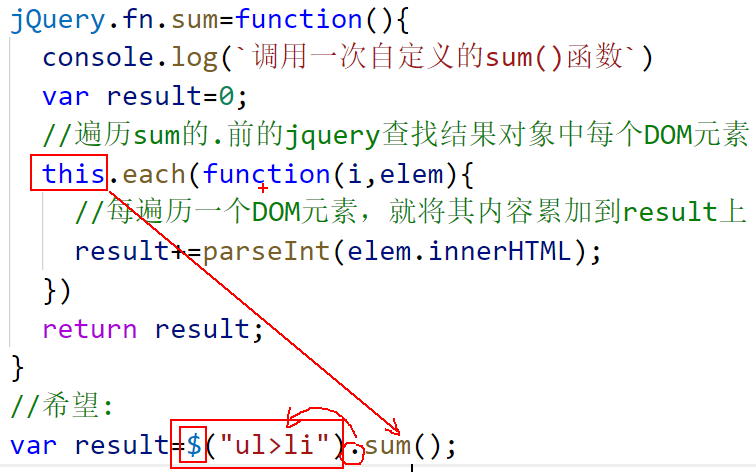
2. 解决: 我们也可以向jQuery的原型对象中添加一个自定义的函数。

3. 如何:

(1). 标准写法: jQuery.prototype.自定义函数=function(){ ... }

(2). 简写: jQuery.fn.自定义函数=function(){ ... }

4. 强调: 因为自定义函数保存在jQuery家的原型对象中，所以，有资格调用自定义函数的.前的主语对象，注定只能是jQuery家的子对象。所以，自定义函数内的this，已经指向调用时.前的jQuery家子对象了，所以，jQuery原型对象方法中的this不需要再$(this)，直接就可以使用jquery家简化版函数。



5. 示例: 为jquery添加sum函数，可对查找结果中所有元素的内容求和

11\_sum.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">    <meta charset="UTF-8">    <title></title>  </head>  <body>    <ul>      <li>85</li>      <li>91</li>      <li>73</li>      <li>59</li>    </ul>    <script src="js/jquery-1.11.3.js"></script>    <script>      //比如，经常需要对找到的所有元素内容求和      jQuery.fn.sum=function(){        console.log(`调用一次自定义的sum()函数`)        var result=0;        //遍历sum的.前的jquery查找结果对象中每个DOM元素        this.each(function(i,elem){          //每遍历一个DOM元素，就将其内容累加到result上          result+=parseInt(elem.innerHTML);        })        return result;      }      //希望:      var result=$("ul>li").sum();      console.log(result);//308    </script>  </body>  </html>  运行结果: |

三. 封装自定义插件:

对应小程序视频: 小程序->在线->JQUERY->day04: 15 jquery ui 封装自定义jquery插件

1. 什么是插件/组件: 拥有专属的HTML+CSS+JS的页面独立的功能区域。

2. 为什么: 项目中很多功能都是重复的。为了避免大量重复的编码，提高开发效率，都要使用插件——重用

3. 何时: 今后只要一个功能在项目中，可能被反复使用，都要封装为插件！

4. 其实, jquery官方提供了一套极其好用的插件！——jQuery UI

(1). 下载: jqueryui.com

(2). 下载之后:

a. 将jquery-ui.css和images文件夹都放在项目的css文件夹下

b. 将jquery-ui.js放在项目的js文件夹下  
 (3). 在网页中引入jqueryui

a. 引入css: <link rel="stylesheet" href="css/jquery-ui.css">

b. 先引入jquery.js: <script src="js/jquery-1.11.3.js">

c. 再引入jquery-ui.js: <script src="js/jquery-ui.js">

(4). 在网页中按插件的要求编写HTML结构和内容，不需要任何class！

(5). 在自定义的js中，使用jquery找到插件的父元素，调用插件函数即可

5. 问题:

(1). jquery ui才用侵入式的方式在后台悄悄自动给元素添加所需的class，程序员不知情！——几乎不可维护！

(2). 不支持移动端和响应式

6. 示例: jquery ui版的手风琴效果：

12\_widget\_accordion.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>    <title> new document </title>    <meta charset="utf-8">    <link rel="stylesheet" href="css/jquery-ui.css">    <script src="js/jquery-1.11.3.js"></script>    <script src="js/jquery-ui.js"></script>  </head>  <body>    <h1>jQueryUI：Widgets —— Accordion</h1>    <!--复制14\_accordion.html中<div>...</div>到此-->    <div id="my-accordion">      <div>《西游记》简介</div>      <div>一个和尚和四个动物的故事: Lorem ipsum dolor sit amet, consectetur adipisicing elit. Similique nulla voluptas velit minus esse voluptatem illum quis magni nihil sint facilis cupiditate nobis quia ab neque. Modi veniam omnis nisi? </div>      <div>《水浒传》简介</div>      <div>105个男人和三个女人的故事: Lorem ipsum dolor sit amet, consectetur adipisicing elit. Omnis provident sapiente aperiam reprehenderit repellat rem magnam vel odio quia harum hic impedit dolorem similique ea est consequatur adipisci at nemo!</div>      <div>《红楼梦》简介</div>      <div>一个男人和一群女人的故事: Lorem ipsum dolor sit amet, consectetur adipisicing elit. Delectus minima quidem aspernatur eligendi optio cupiditate minus nam expedita? Aliquid veritatis doloribus maxime vel dicta illo unde iusto qui quasi doloremque.</div>    </div>    <script>      $("#my-accordion").accordion();    </script>  </body>  </html>  运行结果: |

7. 模仿jquery ui方式，添加自定义插件

(1). 前提: 先用传统的HTML+CSS+JQUERY已经实现了功能。只不过，现在的代码是和页面中其它代码混在一起的，不便于重用！所以，封装自定义插件的过程其实是一个提取代码的过程，而不是从0开始开发的过程。

(2). 第一步: 将插件相关的css提取到一个独立的.css文件中保存

(3). 第二步: 自定义一个插件的函数，添加到jQuery的原型对象中

jQuery.fn.myAccordion=function(){

a. 自动为插件父元素及其子元素添加必须的class——侵入

b. 自动为插件中对应元素绑定事件处理函数

}

8. 使用自定义插件: 和使用jQuery ui插件一样

(1). 拷贝我们自己的插件库文件夹（包含插件.css和插件.js文件）到新项目中  
 (2). 在网页中引入自己插件库的css和js文件

a. 引入css: <link rel="stylesheet" href="插件库文件夹/插件库.css">

b. 先引入jquery.js: <script src="js/jquery-1.11.3.js">

c. 再引入jquery-ui.js: <script src="插件库文件夹/插件库.js">

(4). 在网页中按插件的要求编写HTML结构和内容，不需要任何class！

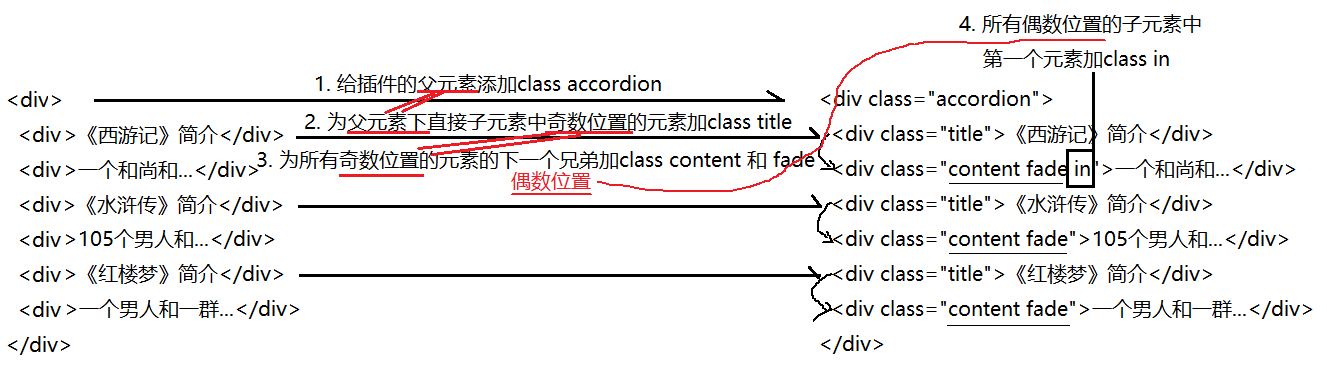
(5). 在自定义的js中，使用jquery找到插件的父元素，调用自定义插件函数即可

9. 示例: 封装自定义手风琴插件

dong-ui/dong-ui.css

|  |
| --- |
| /\*放将来我的插件库中所有插件的css代码\*/  /\*\*\*\*\*\*\*\*\*\*\*\*手风琴\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  .accordion{width:80%; margin:0 auto;}  .accordion>.title{    background:#eee; border:1px soild #aaa;    padding:6px; font-size:1.5em;    font-weight:bold; cursor:pointer;  }  .accordion>.content{    border-left:1px solid #eee;    border-right:1px solid #eee;  }  .accordion>:last-child{    border-bottom:1px solid #eee;  }  .fade{    height:0;    opacity:0;    overflow:hidden;    transition:all .5s linear;  }  .in{    height:84px;    opacity:1;  } |

dong-ui/dong-ui.js



|  |
| --- |
| /\*保存我的插件库中所有插件的函数定义\*/  jQuery.fn.myAccordion=function(){    //a. 自动为插件父元素及其子元素添加必须的class——侵入    //(1). 为当前插件的父元素加class accordion    this.addClass("accordion")    //return this    //(2). 为父元素下直接子元素中奇数位置的元素加class title    .children(":nth-child(2n+1)") //3个奇数位置的元素    .addClass("title")    //return 3个奇数位置的元素    //(3). 为所有奇数位置的下一个兄弟（偶数位置）加class content和fade    .next() //自带遍历: 3个偶数位置的元素    .addClass("content fade")    //return 3个偶数位置的元素    //(4). 为所有偶数位置的元素中第一个元素加class in    .first() //获得任何查找结果中排在第一个位置的一个元素——本例中获得3个偶数位置的元素中的第一个偶数位置的元素。    .addClass("in");    //b. 自动为插件中对应元素绑定事件处理函数    //其实就是将原来已经实现的单击事件处理函数拷贝到此！    $(".accordion").on("click",".title",e=>      $(e.target).next(".content").toggleClass("in")        .siblings(".content").removeClass("in")    );  }  //将来希望:  // 已经是$(...)了   <-this  //$("#my-accordion").myAccordion(); |

14\_accordion.html

|  |
| --- |
| <!DOCTYPE html>  <html>   <head>    <title> new document </title>    <meta charset="utf-8">    <link rel="stylesheet" href="dong-ui/dong-ui.css">   </head>   <body>    <h1>使用“高度动画”实现“手风琴”组件</h1>    <div id="my-accordion">      <div>《西游记》简介</div>      <div>一个和尚和四个动物的故事: Lorem ipsum dolor sit amet, consectetur adipisicing elit. Similique nulla voluptas velit minus esse voluptatem illum quis magni nihil sint facilis cupiditate nobis quia ab neque. Modi veniam omnis nisi? </div>      <div>《水浒传》简介</div>      <div>105个男人和三个女人的故事: Lorem ipsum dolor sit amet, consectetur adipisicing elit. Omnis provident sapiente aperiam reprehenderit repellat rem magnam vel odio quia harum hic impedit dolorem similique ea est consequatur adipisci at nemo!</div>      <div>《红楼梦》简介</div>      <div>一个男人和一群女人的故事: Lorem ipsum dolor sit amet, consectetur adipisicing elit. Delectus minima quidem aspernatur eligendi optio cupiditate minus nam expedita? Aliquid veritatis doloribus maxime vel dicta illo unde iusto qui quasi doloremque.</div>    </div>    <script src="js/jquery-1.11.3.js"></script>    <script src="dong-ui/dong-ui.js"></script>    <script>      //将来希望:      $("#my-accordion").myAccordion();    </script>   </body>  </html>  运行结果: |

四. ajax

对应小程序视频: 小程序->在线->JQUERY->day04: 16 jQuery发送ajax请求$.ajax

1. 从jquery开始，今后所有的函数库或框架都提供了一句话发送ajax请求的方法。

2. jq中:

$.ajax({

url:"服务器端接口地址",

type:"请求类型",

data:{ 参数名: 参数值, ...: ..., ... }, //如果服务器端接口不需要带参数则可以省略data

dataType:"返回值的类型",

success:function(result){//仅当请求响应成功结束后自动调用！

//result收到的是服务器端返回的结果

}

})  
3. 其中:

(1). type: 请求类型，最常用的两个: get或post

(2). dataType: 返回值的类型。因为今后几乎所有服务器端接口都会返回json字符串格式的数据。所以dataType几乎都是"json"。dataType还会自动调用JSON.parse()将返回的json字符串转化为js内存中的数组或对象。

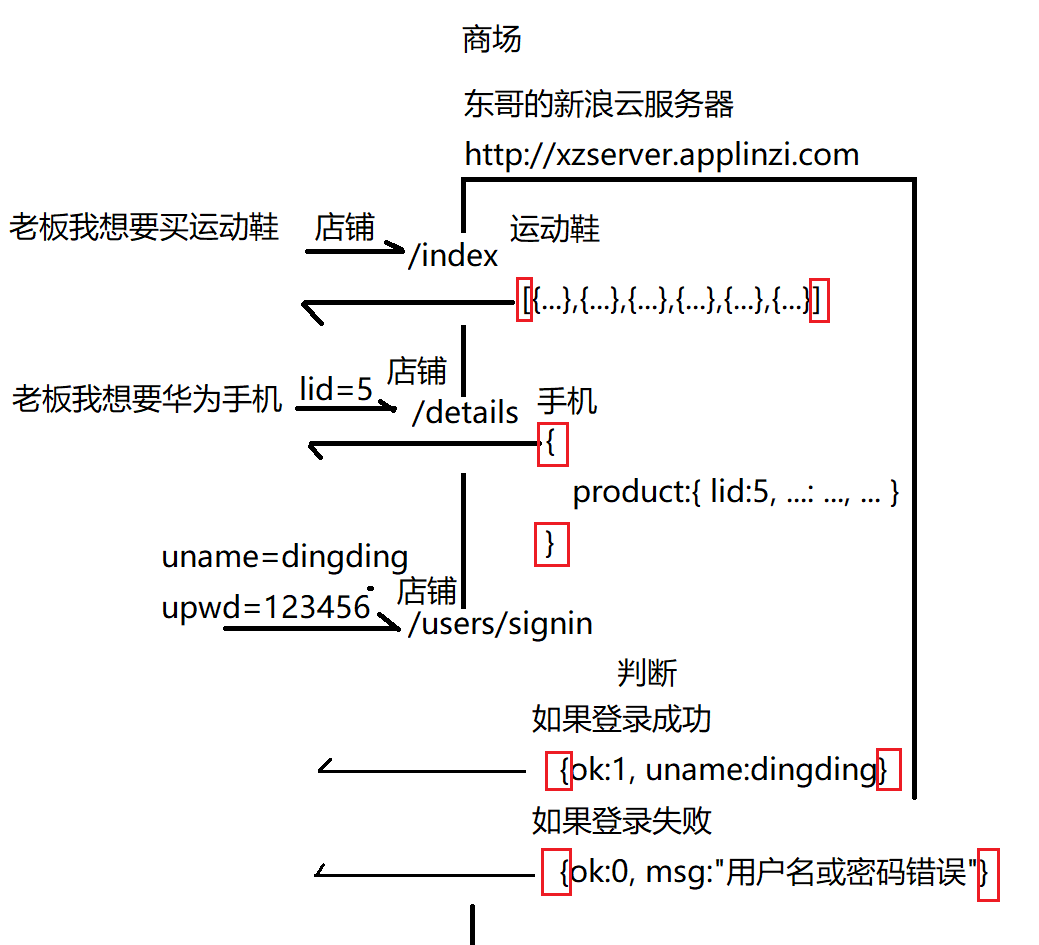
(3). success: 是回调函数，仅在请求响应成功结束后才自动调用

a. success的参数result，可自动接住服务器端返回的结果经过JSON.parse()转化后的直接可用的数组或对象。

b. 因为ajax是异步的，所以，今后，凡是希望在ajax请求结束后才能执行的程序，必须都要放在$.ajax()中的success回调函数内！

4. 示例:

新浪云服务器端接口地址和返回的数据:



15\_$.ajax.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Document</title>  </head>  <body>    <table id="data">      <thead>        <tr>          <th>商品名称</th>          <th>描述</th>          <th>单价</th>        </tr>      </thead>      <tbody>      </tbody>    </table>    <script src="js/jquery-1.11.3.js"></script>    <script>      //必须用live server运行，且端口必须是5500      //1. 去东哥新浪云服务器请求学子商城首页商品列表/index      $.ajax({        url:"http://xzserver.applinzi.com/index",        type:"get",        // data:{},//因为不需要携带参数        dataType:"json",        success:function(result){          console.log(result);          var html="";//定义变量html，准备接每行的html判断          //遍历结果数组中每个商品          for(var p of result){            //每遍历一个商品，就创建一行HTML片段            //同时设置行中每个td的内容为商品对象p的对应属性值。然后把新行的代码添加到HTML变量中            html+=`<tr>              <td>${p.title}</td>              <td>${p.details}</td>              <td>¥${p.price.toFixed(2)}</td>            </tr>`;          }          //循环结束后，将html中保存的所有HTML代码一次性添加到tbody中          //追加到id为data的table下的tbody内          $(html).appendTo("#data>tbody")        }      })    </script>  </body>  </html>  运行结果: |

15\_$.ajax2.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Document</title>  </head>  <body>    <ul>      <li>品名: <span id="title"></span></li>      <li>描述: <span id="subtitle"></span></li>      <li>单价: ¥<span id="price"></span></li>      <li>服务承诺: <span id="promise"></span></li>    </ul>    <script src="js/jquery-1.11.3.js"></script>    <script>      //!!!地址栏: xxx.html?lid=10      //从地址栏获得查询字符串参数      var search=location.search;      //如果有查询字符串参数的话      if(search!==""){        console.log(search);//?lid=10        //                        |        //                 ["?lid","10"]        //                    [0]   [1]        //将search按=切割，然后取第二部分商品编号值        var lid=search.split("=")[1];        //想从东哥新浪云服务器请求5号商品的详细信息        $.ajax({          url:"http://xzserver.applinzi.com/details",          type:"get",          //将从地址栏获得的商品编号值，传到服务器上，作为查询商品详细信息的条件。          data:{lid},//{ lid:lid },          dataType:"json",          success:function(result){            console.log(result);            var {product}=result;            var {title, subtitle, price, promise}=product;            $("#title").html(title);            $("#subtitle").html(subtitle);            $("#price").html(price.toFixed(2));            $("#promise").html(promise);          }        })      }    </script>  </body>  </html>  运行结果: |

15\_$.ajax3.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Document</title>  </head>  <body>    用户名:<input><br/>    密码:<input type="password"><br/>    <button>登录</button>    <script src="js/jquery-1.11.3.js"></script>    <script>      $("button").click(function(){        var uname=$(":text").val();        var upwd=$(":password").val();        $.ajax({          url:"http://xzserver.applinzi.com/users/signin",          type:"post",          data:{ uname, upwd },             //{uname:uname, upwd:upwd}          dataType:"json",          success:function(result){            if(result.ok==1){              alert(`登录成功，欢迎${result.uname}`);            }else{              alert(`登录失败, ${result.msg}`)            }          }        })      })    </script>  </body>  </html>  运行结果: |

五. \*\*\*\*\*跨域:

对应小程序视频: 小程序->在线->JQUERY->day04: 17 jquery跨域cors ...

作业: 高频笔试题 jsonp 跨域

1. 什么是跨域: 一个网站内的网页，去请求并使用了另一个域名下的网站的资源

2. 比如:

<link rel="stylesheet" href="别人网站的css">

<script src="别人网站的js">

<img src="别人网站的图片">

<iframe src="别人网站的网页">

... ...

3. 包括:

(1). 域名不同:

http://www.a.com/网页 -> http://www.b.com/资源

(2). 子级域名不同:

http://hr.tedu.com/网页 -> http://oa.tedu.com/资源

(3). 端口号不同:

http://localhost:5500/网页 -> http://localhost:3000/资源

(4). 协议不同:

http://12306.cn/网页 -> https://12306.cn/资源

(5). 即使同一台机器，IP与域名之间互访也算跨域

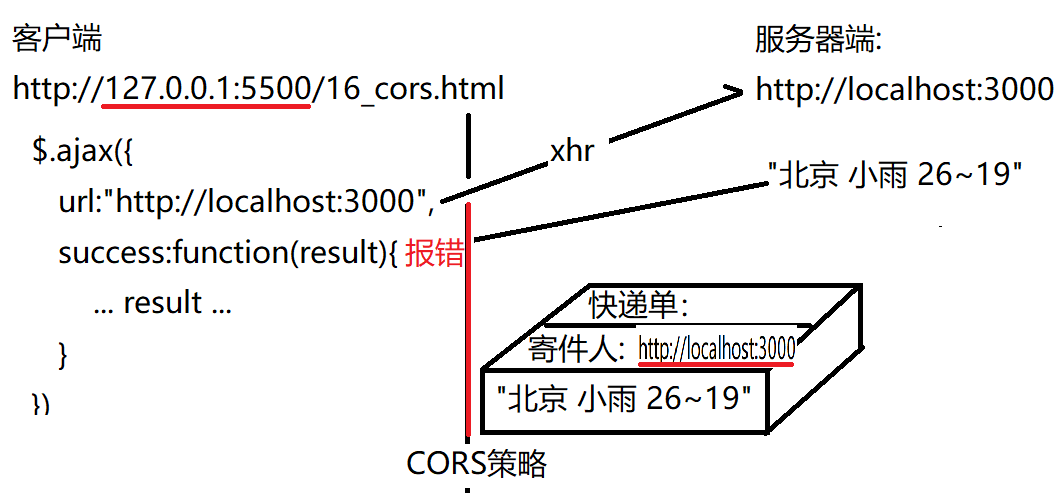
http://localhost/网页 -> http://127.0.0.1/资源

4. 问题: 浏览器允许除xhr对象之外的一切其他手段，发送跨域请求，接收其他网站的资源。唯独禁止xhr对象发送跨域的ajax请求！

5. 报错: Access to XMLHttpRequest at 'http://localhost:3000/' from origin 'http://127.0.0.1:5500' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

翻译: 从http://127.0.0.1:5500 到 http://localhost:3000的XMLHttpRequest请求，已经被CORS策略拦截：在请求回来的资源上，没有设置'Access-Control-Allow-Origin'请求头属性

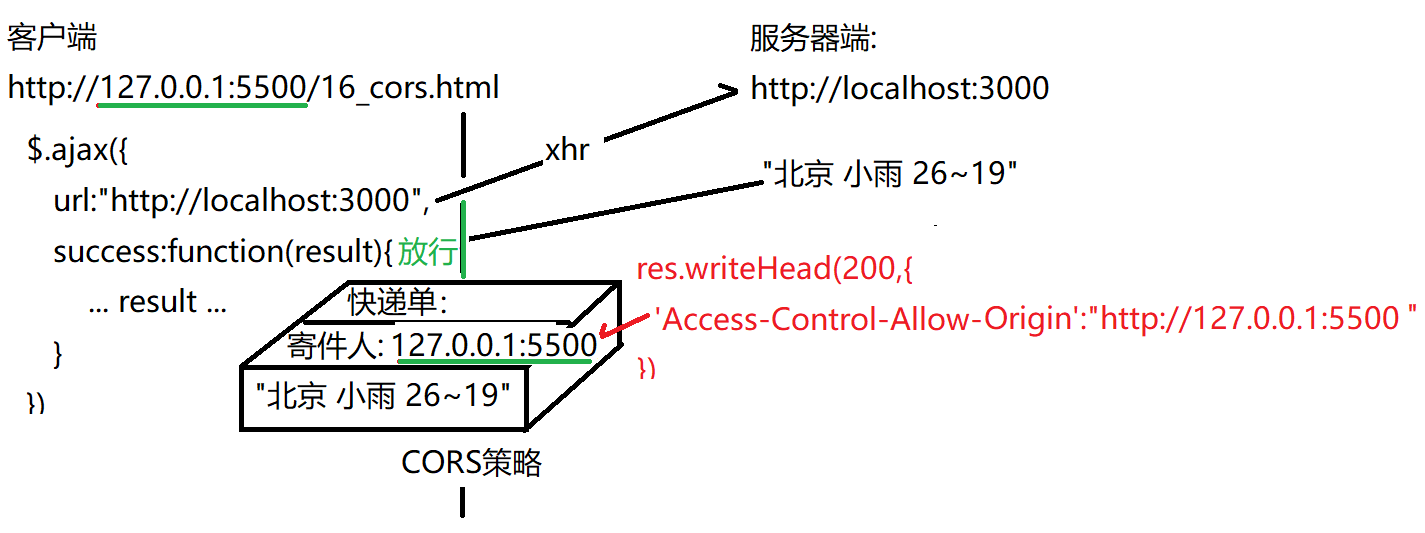
6. 浏览器的CORS策略: 浏览器会检查网页请求回来的所有数据。尤其会检查ajax的xhr请求回来的数据。浏览器只允许来源地址和当前网页所在地址属于同一域名下的数据，进入程序中使用。一旦发现，请求回来的数据来源地址，与当前网页所在地址不一致，则浏览器禁止使用该数据。——同源策略(Cross Origin Resoures Sharing)



7. 解决: 3种:

(1). CORS方式：

a. 请服务器端根据客户端的需要篡改响应结果的寄件人地址，与客户端地址保持一致！



b. 如何:

res.writeHead(200,{

"Access-Control-Allow-Origin":"http://客户端地址:端口号",

... : ...

})。

c. 示例: 使用cors方式解决跨域问题:

16\_server.js

|  |
| --- |
| const http=require("http");  http.createServer((req,res)=>{    var weather="北京 小雨 26~19";    res.writeHead(200,{      "Content-Type":"text/plain;charset=utf-8",      //请服务器端篡改响应结果中的"寄件人地址"与客户端地址保持一致！      'Access-Control-Allow-Origin':"http://127.0.0.1:5500"    });    res.write(weather);    res.end();  }).listen(3000)  //运行:  //右键单击16\_server.js,选在集成终端中打开  //在弹出窗口中运行node 16\_server.js  //打开浏览器，地址栏输入:  //http://localhost:3000  //看到天气预报，说明服务器端正常！  //常见错误:  //Error: listen EADDRINUSE :::3000  //连续点右侧垃圾桶图标,直到关闭终端窗口为止，重新运行 |

16\_cors.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Document</title>  </head>  <body>    <script src="js/jquery-1.11.3.js"></script>    <script>      $.ajax({        url:"http://localhost:3000",        type:"get",        success:function(result){          document.write(`<h1>${result}</h1>`)        }      })    </script>  </body>  </html>  运行结果: |

d. 问题: 如果不同域名地址的多个客户端网页都需要访问同一个服务器端。如何篡改寄件人？

e. 解决: 其实: "Access-Control-Allow-Origin": "\*"

f. 问题: 如果用\*，意为着所有客户端可以不加限制的访问服务器端接口数据，不安全。

(2). JSONP方式: (JSON with Padding)

a. 什么是: 将服务器端要返回的结果，填充进一条可执行的js语句中返回。客户端，不要用ajax发送请求，而用<script>发送请求

b. 如何:

1). 客户端不要用任何ajax方式发送请求，改为用<script src="接口地址">方式发送请求——<script>是允许跨域请求的！

2). 服务器端将要返回的数据，拼接在一条可执行的js语句中，返回一条可执行的js语句

c. 原理:

1). <script src="">向服务器发送请求

2). 服务器返回一条包含数据的可执行的js语句

3). <script>收到可执行的js语句后，立刻自动执行语句。

d. 问题: 要执行的js语句是在服务器端写死的，很可能客户端需要的是不同的操作！

c. 解决:

1). 客户端先在本地定义一个带形参的函数，用来等待将来处理服务器端返回的结果

2). 服务器端返回一条以客户端函数命名的函数调用语句，其中包含要返回的数据作为参数。

d. 原理:

1). 服务器端返回以客户端函数命名的函数调用语句

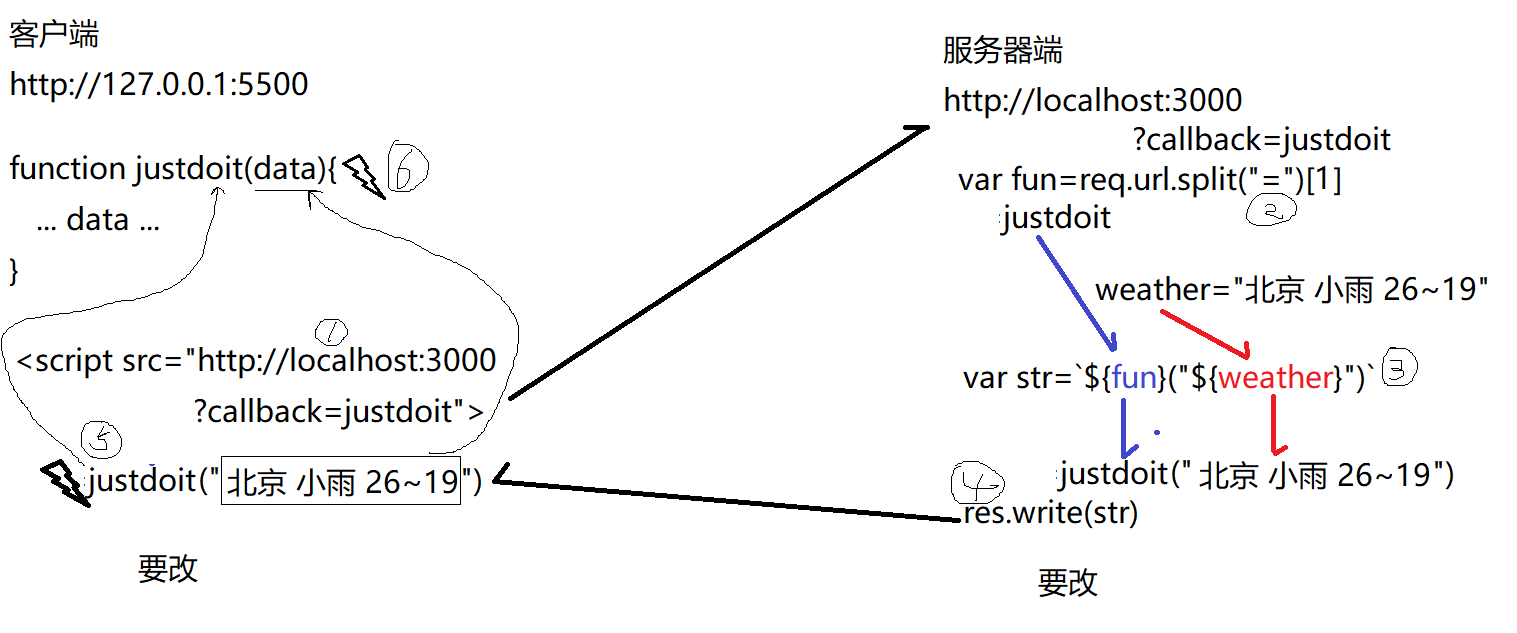
2). <script>接到之后，立刻调用这条语句

3). 刚好客户端内存中之前提前定义了一个同名的js函数，就相当于自动调用该函数执行操作了！

e. 新问题: 函数名是在服务器端写死的，万一修改客户端函数名，服务器端也被迫要同时修改函数名——紧耦合

f. 解决:

1). 客户端在使用<script src="">发送请求时，携带一个提前商量好的参数名和参数值。其中参数值就是客户端本地的函数名 2). 服务器端在接到客户端发来的函数名时，用客户端发来的函数名拼接成一条可执行的函数调用语句，再返回。



g. 示例: 使用jsonp方式跨域:

17\_server2.js

|  |
| --- |
| const http=require("http");  http.createServer((req,res)=>{    //从请求参数中获得函数名    var fun=req.url.split("=")[1];    var weather="北京 小雨 26~19";    res.writeHead(200,{      "Content-Type":"text/plain;charset=utf-8",    });    // var str=`document.write("${weather}")`;    // var str=`tryit("${weather}")`;    var str=`${fun}("${weather}")`    console.log(str);    res.write(str);    res.end();  }).listen(3000)  //运行:  //右键单击16\_server.js,选在集成终端中打开  //在弹出窗口中运行node 16\_server.js  //打开浏览器，地址栏输入:  //http://localhost:3000  //看到天气预报，说明服务器端正常！  //常见错误:  //Error: listen EADDRINUSE :::3000  //连续点右侧垃圾桶图标,直到关闭终端窗口为止，重新运行 |

17\_JSONP.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Document</title>  </head>  <body>    <script>      function justdoit(data){        console.log(`自动调用了justdoit()`)        alert(data);        console.log(data);      }    </script>    <script src="http://localhost:3000?callback=justdoit">      //特点: 接收一条js程序，并立刻执行js程序      //`doit("北京 下雨 26~19")`    </script>  </body>  </html>  运行结果: |

h. 优点: 不需要知道客户端ip，就可实现跨域

i. 缺点: 客户端和服务器端都需要修改，太麻烦了！

(3). http代理/服务器中继: （待续...）

总结: $()共有4种:

1. $("选择器") 查找DOM元素，并包装进jQuery对象中

2. $(DOM元素对象) 不用查找，直接将DOM元素包装进jQuery对象中

3. $(`HTML片段`) 创建新元素

4. $(function(){ ... }) 绑定DOMContentLoaded事件处理函数，在DOM内容加载后就自动提前执行！

总结:

9. 事件绑定:

(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)

12. 添加自定义函数:

a. 添加jQuery.fn.自定义函数=function(){ ... }

b. 调用: $jq对象.自定义函数()  
13. 封装自定义插件:

a. 封装:

1). 将css提取到独立的插件.css文件中

2). 在独立js文件中为jquery原型对象添加自定义插件函数

jQuery.fn.自定义插件函数=function(){

//自动做2件事:

//1. 为插件所在元素及其子元素自动添加class  
 //2. 将原网页中插件所需的事件绑定代码剪切到插件函数中

}

b. 使用自定义插件:

1). 引入jquery.js，插件.js，插件.css

2). 按插件要求编写HTML内容，不用加任何class

3). 在自定义js中，查找插件所在元素，调用自定义插件函数

14. 发送ajax请求:

$.ajax({

url:"服务器端接口地址",

type:"get或post",

data:{ 参数名: 参数值, ... },

dataType:"json",

success:function(result){

//result就是服务器端返回的结果

//不用自己调JSON.parse()

}

})

15. 跨域:

a. 同源策略(CORS): 浏览器只允许当前网页中的ajax请求使用自己网站的资源，禁止ajax请求使用其他网站的资源。

b. 解决: cors方式：只靠服务器端就能完成:

res.writeHead(200,{

... ...  
 "Access-Control-Allow-Origin": "http://客户端网页所在网址"  
})

c. 结果: 服务器端将响应结果中的寄件人地址伪装成和客户端网页所在地址一致，就可骗过浏览器cors策略的检查，让ajax顺利使用其他网站服务器端返回的结果数据了。

d. 问题: 服务器端必须知道客户端网站的域名或ip地址，受局限。

e. 解决: JSONP方式，服务器端无需知道任何客户端的IP或域名，没有局限。  
 1). 客户端提前定义处理数据的函数: function 函数名(形参){ ... }

2). 客户端不要用ajax或xhr发送请求，改为用<script src="接口地址">发送请求

3). 服务器端将要返回的数据，拼接进一条字符串格式的js函数调用语句中，然后返回整条包含数据的函数调用语句。函数名与客户端提前定义的处理函数名保持一致

4). 结果: <script>收到服务器端返回的函数调用语句后，立刻执行，自动调用第一步已经定义好的处理函数并将拼接在函数调用语句中的数据，传给处理函数的形参变量。

今日对应小程序视频:

小程序->在线->JQUERY->day03 5. 类数组对象操作

小程序->在线->JQUERY->day04:

14 添加jQUery自定义函数...

15 jquery ui 封装自定义jquery插件  
 16 jQuery发送ajax请求$.ajax

17 jquery跨域cors ...

作业: 高频笔试题 jsonp 跨域

作业:

1. 复习今日小程序问题清单: 小程序->首页->JQUERY->day04

2. 必须复习:

小程序->在线->JSCORE->day05 5. 访问器属性

小程序->在线->JSCORE->day07 作业: ES6中关于对象中属性和方法的简写

小程序->在线->JSCORE->day07 6和7 解构

小程序->在线->DOM->day01 作业: 扩展笔试题: 递归遍历一个父元素所有后代元素

3. 预习: 小程序->在线->VUE->day01

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