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-> day03、day04

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-> day03、day04

(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->day04

作业: 高频笔试题 jsonp跨域

16. jquery跨域 cors 同源策略 express脚手架

项目: 学子商城项目 1~4

3. 今日课程中包含的英语单词的翻译：小程序->单词->JQUERY->day03、04

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6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

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 4. 类数组对象操作

1. 问题: jQuery中所有查找结果都是类数组对象，但是js语言非常歧视类数组对象。虽然长的像数组，但是无法使用数组家好的函数！就会限制jQuery查找结果发挥更大的作用.

2. 解决: jQuery中模仿着数组家最常用的两个函数forEach和indexOf，专门定义了两个新函数each()和index()，专门供jQuery查找结果的类数组对象使用！

3. 遍历查询结果中每个DOM元素: each():模仿数组家的forEach:

(1). 回顾: 数组家的forEach

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

2). 依次遍历arr数组中每个元素，每遍历一个元素，就自动执行回调函数，对每个元素执行相同的操作

(2). jq中: 新增each()

1). $(...).each(function(i, elem){

//i 获得当前正在遍历的下标位置

//elem 获得当前正在遍历的一个DOM元素

})

(3). 示例: 遍历ul下每li，并按要求修改符合条件的li

day03 each index/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"];  console.log(arr);  arr.forEach(function(elem){  console.log(`${elem} - 到！`)  })  //请给每个不足60分的成绩+10分，并将超过90分的成绩用绿色背景标识出来  console.log($("ul>li"));//爹是Object  //错误: 类数组对象不能用数组家的函数  //$("ul>li").forEach(function(elem){  //正确:  $("ul>li").each(function(i,elem){  //先将当前正在遍历的dom元素elem，转为jquery对象  var $elem=$(elem);  //再获得元素的内容，转为整数  var score=parseInt($elem.html());  // 副本 按值传递  //如果elem的内容<60，就+10分  if(score<60){  score+=10;//修改副本，页面是不变的  //必须将新值再放回去，页面才能改变  $elem.html(score);  }else if(score>=90){//否则如果elem的内容>=90，就让elem变绿  $elem.css("background-color","lightGreen")  }  })  </script>  </body>  </html>  运行结果: |

4. 获取元素在查找结果中的位置：模仿数组家的indexOf()

(1). 回顾数组家indexOf()

a. var i=arr.indexOf(要找的元素值)

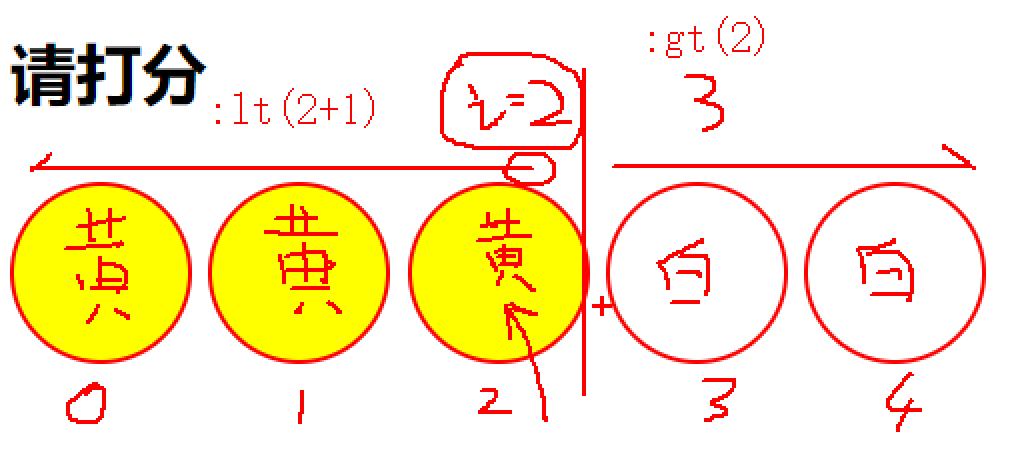
b. 在数组arr中查找“要找的元素值”出现的下标位置。如果找不到返回-1

(2). jq中: 新增index()

a. var i=$(...).index(DOM元素)

b. 查找一个指定DOM元素在$(...)查找结果中的下标位置

(3). 示例: 五星评价



day03 each index/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indexOf  var arr=["亮亮","然然","东东"];  //想查找然然的位置  var i=arr.indexOf("然然");  console.log(`然然在${i}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  //DOM 4步  //1. 查找触发事件的元素  //本例中: ul下的所有li都能单击  $("ul>li")  //2. 绑定事件处理函数  .click(function(){  //3. 查找要修改的元素  //4. 修改元素  //先获得当前点击的li在所有li中的下标位置i  var i=$("ul>li").index(this)  //再将<i+1位置的所有li背景变为黄色  //错误: i是变量！直接放在字符串中，就不是变量了！  //$(`ul>li:lt(i+1)`)  //正确: 用模板字符串将变量i拼接到选择器中  $(`ul>li:lt(${i+1})`).css("background-color","yellow");  //再见>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: |

对应小程序视频: 小程序->在线->JQUERY->day04 14. 添加jQuery自定义函数 封装jquery自定义插件

二. 添加自定义函数：

1. 问题: 我常用的功能，jquery家没有提供

2. 解决: 自定义一个函数，添加到jQuery家原型对象中

3. 如何:

jQuery.prototype.自定义函数=function(形参列表){

... ...

}

4. 问题: 如何获得将来调用这个公共函数的.前的jQuery查找结果对象？

5. 解决: this:

因为将来: $(...).自定义函数()

所以: this->.前的jquery查找结果对象$(...)

所以: jQuery原型对象方法中的this，不用再$(this)的。因为已经是jQuery子对象了。

6. 示例: 为jQuery家添加对查找结果中所有DOM元素内容求和的sum()函数

11\_sum.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head lang="en">  <meta charset="UTF-8">  <title></title>  <script src="js/jquery-1.11.3.js"></script>  <script>  //在jQuery家原型对象中添加共用的函数sum  jQuery.prototype.sum=function(){  console.log(`调用了一次我自定义的sum函数`)  //遍历sum.前的查找结果中的每个dom元素的内容，并累加求和  var result=0;  //问题: 如何获得将来调用这个公共函数的.前的查找结果对象？！  //将来: $("ul>li").sum()  // this->$("ul>li")  for(var i=0;i<this.length;i++){  //因为this是$("ul>li")，是类数组对象，其中包含多个DOM元素li对象  //this[i]取出的应该是一个DOM的li对象  result+=parseInt(this[i].innerHTML)  }  return result;  }  </script>  </head>  <body>  <ul>  <li>85</li>  <li>91</li>  <li>73</li>  <li>59</li>  </ul>  <script>  //比如: 我经常需要对找到的所有元素内容求和！  // jquery家子对象  var result=$("ul>li").sum()  console.log(result);  </script>  </body>  </html>  运行结果: |

三. 封装自定义插件:

1. 什么是插件/组件: 包含专属的HTML，CSS，JS的独立的可重用的页面功能区域

2. 为什么: 重用！

3. 何时: 如果页面中一个独立的功能区域在项目中多个位置被反复用到，都要封装为插件，再反复使用插件！

4. 其实jquery官方提供了一套极好用的插件: jQuery UI

(1). jqueryui.com

(2). 如何使用: 仅手风琴效果举例:

a. 在页面中先引入jquery，再引入jqueryui的css和js

<script src="js/jquery-1.11.3.js">

<script src="js/jquery-ui.js">

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

b. 按插件要求编写HTML内容(同bootstrap)

不同点: 使用jqueryUI时，HTML元素上不用加任何class！

c. 在自定义js中，用jquery找到插件的父元素，调用jqueryui提前定义好的插件函数即可！

(3). 问题: 只有pc端，没有移动端，且不支持响应式！——jQuery几乎退出历史舞台，被bootstrap替代了！

(4). 示例: 使用jqueryui插件实现手风琴效果

14\_widget\_accordion.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title> new document </title>  <meta charset="utf-8">  <script src="js/jquery-1.11.3.js"></script>  <script src="js/jquery-ui.js"></script>  <link rel="stylesheet" href="css/jquery-ui.css">  </head>  <body>  <h1>jQueryUI：Widgets —— Accordion</h1>  <!--以下HTML从14\_accordion.html中复制过来的，且去掉了所有的class-->  <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： |

5. 模仿jqueryui风格，封装自定义jquery插件:

(0). 前提: 已经用传统的HTML，css和js实现了插件的功能，只不过和别的网页内容混在一起，没有独立出来，无法重用而已！

所以，封装插件的过程，其实只是一个提取HTML，CSS和js的过程。而不是从0开始制作一个功能！

(1). 先创建一个独立的css文件，将原页面中插件相关的css剪切到独立的css文件中保存。

(2). 再创建一个独立的js文件，在独立的js中向jquery原型对象中添加一个自定义插件函数，做2件事:

a. 为要应用插件的元素及其子元素自动添加必须的class

b. 自动绑定事件处理函数:

因为原页面中已经实现了插件的功能，所以只要单纯将原页面中的事件绑定代码剪切到插件函数中，放在自动添加class之后即可！

6. 如何在HTML页面中使用自定义jquery插件: 同使用jqueryui插件完全一样！

7. 总结: 从此如果其他项目也想用这个插件:

(1). 只要将包含插件的css和js的文件夹/压缩包拷贝到新项目

(2). 在新项目的网页中引入jquery、插件的js和css

(3). 调用一次插件函数即可！

8. 示例: 封装一个手风琴插件

my-ui/my-ui.css

|  |
| --- |
| .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;  } |

my-ui/my-ui.js

|  |
| --- |
| jQuery.prototype.myAccordion=function(){  console.log(`调用了自定义的myAccordion()函数`)  //1. 为要应用插件的当前元素及其子元素自动添加class  //问题: 如何获得当前要应用插件的元素?  //解决：this->将来要应用插件的.前的元素  //因为this->$("#my-accordion")，已经是jQuery子对象，所以不用$(this);  var $parent=this;//$parent->id为my-accordion的父元素div  //为父元素div本身添加class accordion  $parent.addClass("accordion")  //为父元素下所有偶数位置的元素加title class  .children(":even").addClass("title")  //return 所有偶数位置的元素  //为每个偶数位置的元素的下一个兄弟元素加content和fade class  .next().addClass("content fade")  //return 所有奇数位置的元素  //为所有奇数位置的元素中第一个元素加class in  .first().addClass("in");  //2. 自动绑定事件  $(".accordion").on("click",".title",e=>  $(e.target).next(".content").toggleClass("in")  .siblings(".content").removeClass("in")  );  }  //将来: 比如: id为my-accordion的div想应用手风琴插件  //按jquery ui做法:  //$("#my-accordion").myAccordion()  //找到插件父元素 调用自定义插件函数 |

14\_accordion.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title> new document </title>  <meta charset="utf-8">  <script src="js/jquery-1.11.3.js"></script>  <script src="my-ui/my-ui.js"></script>  <link rel="stylesheet" href="my-ui/my-ui.css">  </head>  <body>  <h1>使用“高度动画”实现“手风琴”组件</h1>  <!--按插件规定编写HTML内容，一个class都不要加！-->  <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>  //查找要应用插件的父元素，只需要调用自定义的插件函数  $("#my-accordion").myAccordion();  //2件事:  //1. 自动添加class  //2. 自动绑定事件  </script>  </body>  </html>  运行结果: |

四. ajax

对应小程序视频: 小程序->在线->JQUERY->day04 15. jquery发送ajax请求$.ajax

1. jquery中已经封装好了简化版的一句话发送ajax请求的函数

2. 固定用法:

$.ajax({

url:"服务器端接口地址",

type:"get或post",

data:{ 变量: 值, 变量: 值 }, //发送到服务器端的参数变量和值

dataType:"json", //因为几乎所有服务器端返回的结果都是json字符串，都必须反复调用JSON.parse()将json字符串，转为js中的对象和数组，程序才能用。所以，发送ajax请求时，只要执行dataType参数值为json，$.ajax就会自动调用JSON.parse()将返回的json字符串，转化为对象或数组

.

. 延迟

.

//相当于onreadystatechange=function(){ ... }

success: function(result){ //只要请求响应成功，成功获得服务器端返回的结果时，自动调用success回调函数。并自动将服务器端返回的结果，经过JSON.parse()后翻译为对象，交给result变量。

//在success函数中，就可使用result来获得服务器端返回的结果执行后续操作

}

})

3. 示例: 使用ajax函数发送多种请求

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>  <script src="js/jquery-1.11.3.js"></script>  </head>  <body>  <script>  //想从东哥新浪云服务器的/index接口获得学子商城首页4个商品  //服务器端接口地址: http://xzserver.applinzi.com/index  $.ajax({  url:"http://xzserver.applinzi.com/index",  type:"get",  dataType:"json",  success:function(result){  console.log(result);  }  });  //想从东哥新浪云服务器的/details接口获得5号商品的详细信息  //服务器端接口地址:  // http://xzserver.applinzi.com/details  // 参数: lid=商品编号  $.ajax({  url:"http://xzserver.applinzi.com/details",  type:"get",  data:{ lid:5 },  dataType:"json",  success:function(result){  console.log(result);  }  });  //想利用东哥新浪云服务器的/signin接口登录学子商城  //服务器端接口地址:  //http://xzserver.applinzi.com/users/signin  //post类型:  //参数: uname=用户名&upwd=密码  // dingding 123456  $.ajax({  url:"http://xzserver.applinzi.com/users/signin",  type:"post",  data:{ uname:"dingding", upwd:"123456" },  dataType:"json",  success:function(result){  //result两种情况:  //如果用户名密码正确: {ok:1, uname:"dingding"}  //如果用户名密码错误: {ok:0, msg:"用户名或密码不正确"}  console.log(result);  if(result.ok==1){  document.write(`<h1>Welcome ${result.uname}</h1>`)  }else{  document.write(`<h1 style="color:red">${result.msg}</h1>`)  }  }  })  </script>  </body>  </html>  运行结果: |

五. \*\*\*\*\*跨域:

对应小程序视频: 小程序->在线->JQUERY->day04 16. jQuery跨域cors同源策略...

1. 什么是: 一个网站下的网页中请求了别的网站的资源！

2. 比如:

<link rel="stylesheet" href="别的网站的css">

<script src="别的网站的js">

<img src="别的网站的图片">

<a href="别的网站的网页">

<iframe src="别的网站的网页">

3. 包括:

网页来源 请求 要请求的资源所在的网址

(1). 域名不同: www.a.com -> www.b.com

(2). 子域名不同: oa.tedu.cn -> hr.tedu.cn

(3). 端口号不同: localhost:5500 -> localhost:3000

(4). 协议不同: http://12306.cn -> https://12306.cn

(5). 同一台主机: http://localhost -> http://127.0.0.1

域名 ip地址

4. 问题: 浏览器禁止ajax发送跨域请求！

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: 从源头'http://127.0.0.1:5500'到目标'http://localhost:3000/'的xhr的访问被CORS策略阻止了。因为响应回来的资源中没有设置Access-Control-Allow-Origin响应头

5. 示例: 抛出跨域错误:

16\_server.js

|  |
| --- |
| const http=require("http");  http.createServer(function(req,res){  var weather=`晴 21~31度`;  res.writeHead(200,{  "Content-Type":"text/plain;charset=utf-8"  });  res.write(weather);  res.end();  }).listen(3000)  如何运行  1. 右键单击16\_server.js，选择"在终端中打开"或"在命令行中打开"  2. 等待打开的窗口中显示 路径> 输入node 16\_server.js 回车 启动服务器端  3. 打开浏览器，地址栏输入http://localhost:3000  看到晴 21~31度,说明服务器端启动成功！  如果启动服务器报: listen xxxxx :: 3000错误  原因: 同时开了多个命令行窗口，都监听3000端口，发生端口冲突  解决: 连续点命令行窗口右上角垃圾桶图标，关闭所有命令行窗口    然后再从第一步开始运行。始终保持命令行窗口只开一个 |

16\_client.html

右键单击16\_client.html，选择"open with live server"

F12->console看到错误！

|  |
| --- |
| <!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>今日北京: ${result}</h1>`)  }  })  </script>  </body>  </html>  运行结果: |

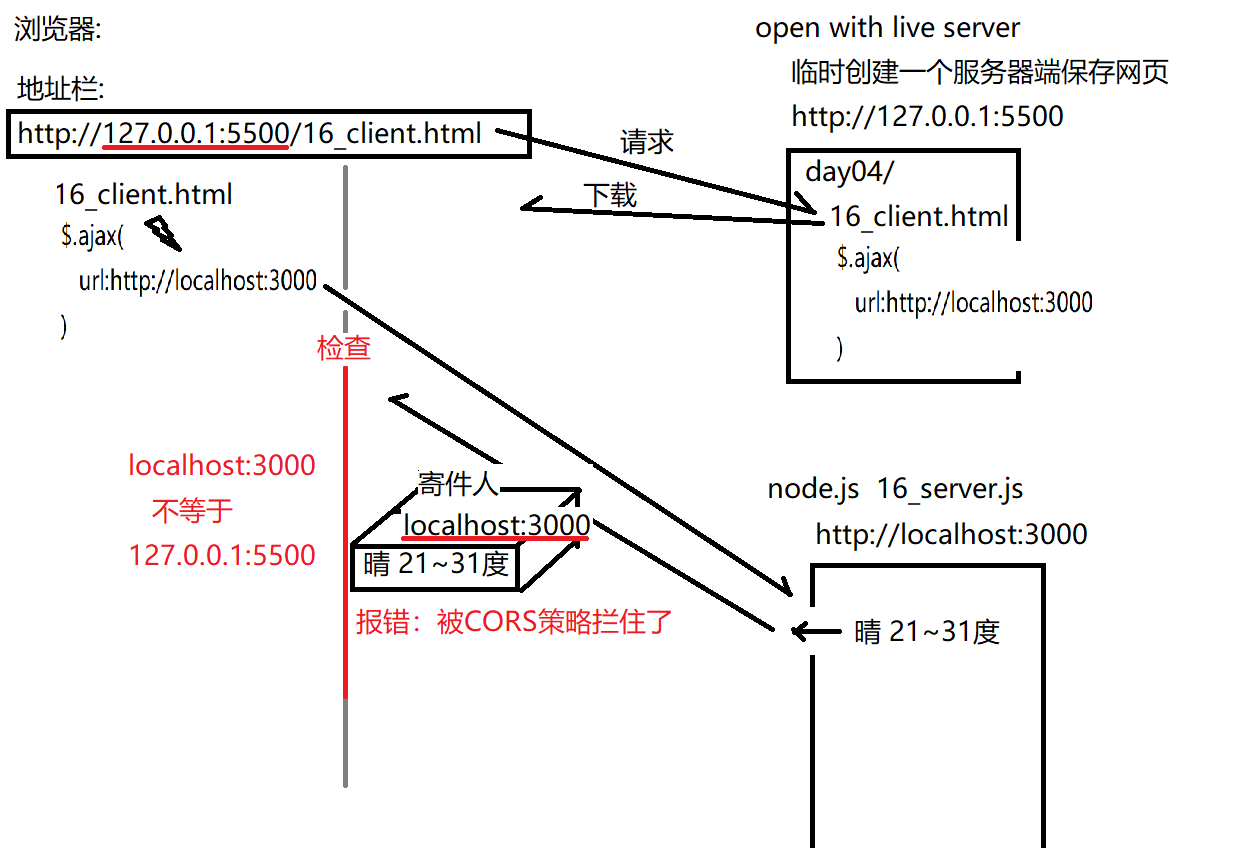
6. 跨域错误的原因: 浏览器都有一个同源策略

(1). 什么是同源策略: 当网页中发送ajax请求时，浏览器会检查ajax获得的响应结果和当前网页是否来自于同一个域名下的服务器。

a. 如果浏览器发现ajax获得响应结果和当前网页不是来自于同一个域名下，则不允许网页使用获得的返回结果

b. 只有浏览器发现ajax获得响应结果和当前网页来自于同一个域名下，才允许网页使用ajax返回的结果！

(2). 比如:



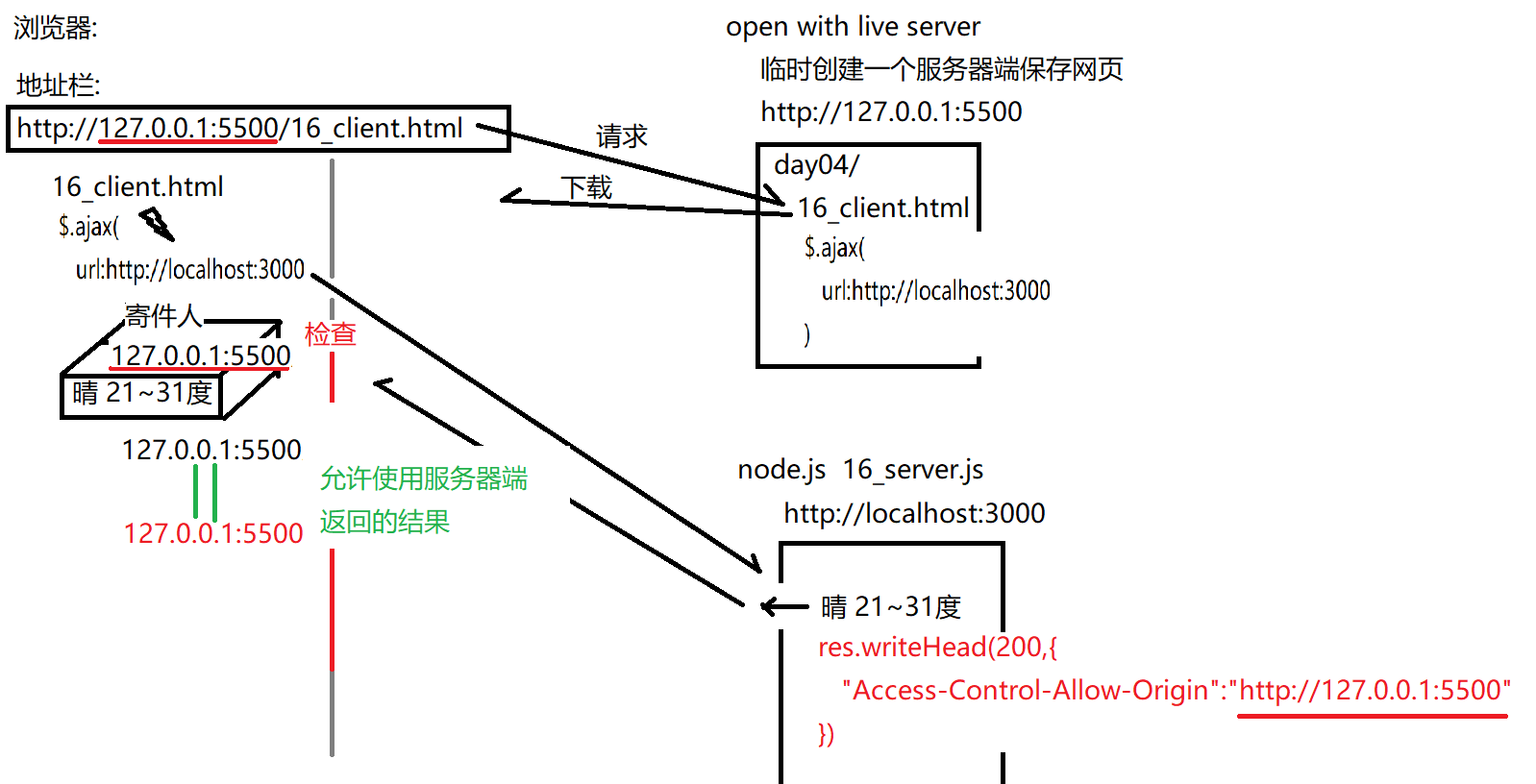
(3). 同源策略: Cross Origin Resources Sharing

跨 不同源头 资源 共享

只有网页来源和响应结果来源相同时才允许使用响应结果

7. 解决: 骗

修改服务器端代码的响应头中的Access-Control-Allow-Origin属性为客户端网页所在地址

8. 示例: 修改服务器端允许http://127.0.0.1:5500的网页跨域使用服务器端数据

修改16\_server.js

|  |
| --- |
| const http=require("http");  http.createServer(function(req,res){  var weather=`晴 21~31度`;  res.writeHead(200,{  "Content-Type":"text/plain;charset=utf-8",  "Access-Control-Allow-Origin":"http://127.0.0.1:5500"  });  res.write(weather);  res.end();  }).listen(3000)  运行结果:  重新运行服务器端  重新用live server打开16\_client.html  看到天气预报，说明成功！  **今日北京: 晴 21~31度** |

9. 问题: 一个服务器端同时包含很多个接口程序，如果每个接口程序都要重复写这么长一坨代码，才能跨域——极其不便于维护？

10. 解决: 各种服务器端语言nodejs, java,...都提供专门支持跨域的中间件程序，仅以nodejs举例:

(1). 服务器端项目安装cors模块:

npm i -save cors

结果: 服务器端项目/node\_modules/cors文件夹

(2). 在服务器端项目主程序app.js中，先引入cors模块，再配置cors中间件

const cors=require("cors");

//在app = express();之后

//配置cors中间件

app.use(cors({

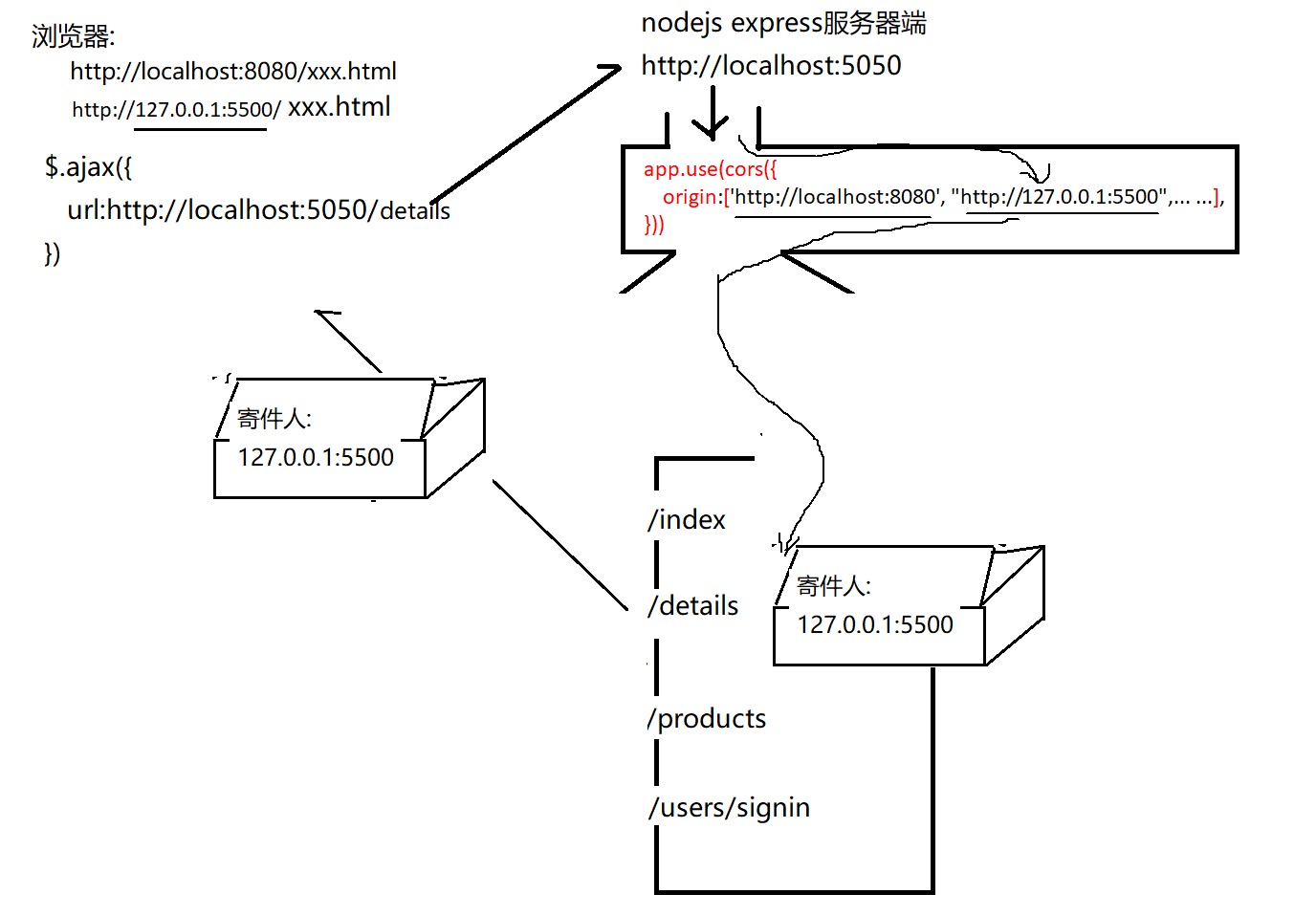
origin:['http://localhost:8080', "http://127.0.0.1:5500",... ...],

VUE脚手架 live server

//希望跨域的客户端网页地址

}))

(3). 结果: 凡是这台服务器的所有接口都可支持origin数组中所有客户端网页的跨域请求。



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)

8. 动画:

(1). 简单动画: 3种固定效果，9个函数

a. 显示隐藏:

$元素.show(动画持续时间毫秒数, function(){ 动画结束后自动执行 })

$元素.hide(动画持续时间毫秒数, function(){ 动画结束后自动执行 })

$元素.toggle(动画持续时间毫秒数, function(){ 动画结束后自动执行 });

b. 上滑下滑:

$元素.slideUp(动画持续时间毫秒数, function(){ 动画结束后自动执行 })

$元素.slideDown(动画持续时间毫秒数, function(){ 动画结束后自动执行 })

$元素.slideToggle(动画持续时间毫秒数, function(){ 动画结束后自动执行 });

c. 淡入淡出:

$元素.slideUp(动画持续时间毫秒数, function(){ 动画结束后自动执行 })

$元素.slideDown(动画持续时间毫秒数, function(){ 动画结束后自动执行 })

$元素.slideToggle(动画持续时间毫秒数, function(){ 动画结束后自动执行 });

(2). 万能动画:

$元素.animate({

css属性: 目标值,

... : ...

}, 动画持续时间毫秒数, function(){动画结束后自动执行})

(3). 停止动画: $元素.stop()

(4). 查找正在播放动画的元素: jquery新增选择器: :animated

9. 类数组对象操作:

(1). 遍历查找结果中每个DOm元素: $(...).each(function(i, elem){ ... })

(2). 查找一个元素在查找结果中的下标位置: $(...).index(要找的DOm元素)

10. 添加自定义函数:

jQuery.prototype.自定义函数=function(形参列表){

this->将来调用这个自定义函数的.前的jQuery子对象

this不用$(this)

}

将来:

$(查找结果).自定义函数()

11. 封装自定义插件:

(1). 创建自定义插件:

a. 先创建独立css文件，将原网页中插件相关的css，剪切到独立的css文件中

b. 再创建独立js文件，为jquery原型对象添加一个自定义插件函数。插件函数中做2件事

1). 为找到的HTML元素自动添加class

2). 为找到的元素自定绑定事件处理函数

(2). 使用插件:

a. 将插件的css和js文件拷贝到项目中

b. 在页面中引入jquery的js和插件的js以及插件的css

c. 在页面中按插件要求编写HTML内容和结果，不需要加class

d. 在自定义js中，查找插件父元素，调用插件函数

12. ajax

$.ajax({

url:"服务器端接口地址",

type:"get或post",

data:{ 变量: 值, 变量: 值 }, //发送到服务器端的参数变量和值

dataType:"json",

success: function(result){

//可使用result来获得服务器端返回的结果执行后续操作

}

})

13. 跨域:

(1). 浏览器同源策略: 浏览器会检查ajax获取的服务器端响应结果的来源地址是否与当前网页来源地址相同。只有相应结果来源于当前网页来源相同，才允许使用响应结果。否则，不允许使用。

(2). 如何跨域:

a. 一个接口:

res.writeHead(200,{

... ...

"Access-Control-Allow-Origin":"要伪装的客户端地址"

})

res.write(要返回的数据)

res.end();

b. nodejs express项目所有接口都要跨域:

1). 安装cors: npm i -save cors

2). app.js中:

const cors=require("cors")

//在var app=express()之后

app.use(cors({

origin:['http://localhost:8080',"http://127.0.0.1:5500",...],

[]中放的是所有可能要伪装的客户端网页地址

}))

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->day03 4. 类数组对象操作

小程序->在线->JQUERY->day04 14. 添加jQuery自定义函数 封装jquery自定义插件

15. jquery发送ajax请求$.ajax

16. jQuery跨域cors同源策略...

作业:

1. 复习今日小程序问题清单(小程序->在线->JQUERY->day03、04)，预习明日小程序问题清单(小程序->在线->VUE->day01)

2.（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