提前准备:

1. 今天的历届师兄师姐问过的经典的高频的问题清单都已经整理在小程序->首页->JQUERY->day04里

2. 今天的课程中所需的单词列表都已经整理在小程序->单词-> JQUERY ->day04里

3. 今天的重点视频、扩展高频鄙视题视频已经整理在小程序->在线-> JQUERY ->day04里

如果不知道问什么: 谁问的越多！谁培训费花的越值！

1. 笔记中或画图中那句话或者那个位置不明白，立刻问！

2. 案例中哪行代码或注释不明白，立刻问！

3. 课下自己做项目，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！——公众号: 前端大全 定期总结大厂笔试题

5. 看小程序问题清单时，遇到看不懂的问题，立刻问！

每个案例三遍:

1. 连注释+代码抄一遍

2. 保留注释，删掉代码，自己试着填回来

3. 删除注释和代码，自己使用用人话把注释写回来！——才算自己会！

正课:

1. 添加自定义函数

2. 封装自定义插件

3. ajax

4. \*\*\*\*\*跨域

一. 添加自定义函数:

1. 何时: 只要想用的函数，jquery没有提供，就可向jquery中添加自定义函数

2. 如何: 其实就是向jquery的原型对象中添加一个新的共有函数

jQuery.fn=jQuery.prototype

//jQuery.prototype.新函数=function(){

jQuery.fn.新函数=function(){

}

3. 强调: 原型对象中的this->将来调用这个函数的.前的某个jquery类型的子对象。因为已经是jQuery类型子对象了，所以不用再$(this)，就可this.jq简化版函数()。

4. 示例: 为jquery添加求和的函数sum

|  |
| --- |
| <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 src="js/jquery-3.2.1.js"></script>  <script>  jQuery.prototype.sum=function(){    var total=0;    //jquery-1.11.3不支持es6语法的。    //$("ul>li")         li    // this.each(function(i,li){    //   total+=parseInt(li.innerHTML)    // })    //jquery从3.x开始支持es6语法    for(var li of this){      total+=parseInt(li.innerHTML)    }    return total;  }  </script>  <script>  //希望  console.log($("ul>li").sum())  </script>  </body> |

二. 封装自定义插件

1. 什么是插件: 拥有专属的HTML，css和js的可重用的独立的页面功能区域

2. 为什么: 重用！提高开发效率

3. 何时: 今后只要项目中一个页面中独立的功能区域，可能被反复使用，都要封装为插件。

4. 其实jquery官方提供了一套很好用的插件: jQuery UI

(1). 官网: jqueryui.com

(2).下载jqueryui的压缩包

a. 将压缩包中的jquery-ui.js放入项目的js文件夹中

b. 将压缩包中的jquery-ui.css和images文件夹放入项目的css文件夹中

(3). 如何:

a. 在网页中按插件要求编写HTML代码(不需要加任何class)

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

c. 先引入jquery.js，再引入jqueryui的js

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

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

d. 在定义<script>中，先用jquery查找到插件HTML片段的父元素，对父元素调用一次jqueryui的插件函数，立刻效果就实现了！

(4). 原理: jqueryui的插件函数会根据自己的需要自动悄悄的为对应元素添加所需的class和属性。

(5). 缺点:

a. 几乎不可维护！

b. 仅支持PC端，不支持响应式和移动端

(6). 示例: 使用jqueryUI实现手风琴效果：

|  |
| --- |
| <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>    <!--从14\_accordion.html中复制<div class="accordion">到此，并删除元素上所有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> |

5. 封装jqueryui风格的自定义插件：

(1). 前提: 已经在页面中用HTML，css和js实现了插件的效果，只不过暂时没有独立出来而已。还和别的功能的代码混为一谈。不便于重用。——所以封装自定义插件不是从0开始创造一个插件的过程。仅仅是将现有插件代码提取出来单独存放和使用的过程

(2). 第一步: 先将插件涉及的css，提取到一个独立的.css文件中保存

a. 强调: 一定要避免插件/组件间样式冲突——高频笔试题

b. 问题: 将来一个插件库中都是集中保存多个插件的css内容，所以多个插件中可能包含相同的class名的样式定义，比如:

1). 手风琴插件可能定义了样式类: .in{ height:100px }

2). 对话框插件可能也定义了样式类: .in{ width:300px; height:200px }

3). 当以上两个样式类被集中放在一个插件库的.css文件中，就会发生冲突(覆盖，合并)！

c. 解决:

1). 不好解决办法: 改名。既被迫需要记很多类名，不便于使用，又在定义类名时没有那么多词可用！

2). 好的解决方法: 使用完整的选择器查找路径

i. 组件的父级元素上，应该先定义一个以组件名命名的尽量和其他组件不重复的class名

ii. 所有这个插件内的子元素上样式的选择器，都必须从这个父元素的统一class名开始向下查找

iii. 优点: 子元素的class名可以随便起名，而不用担心冲突

(2). 在独立的.js文件中向jquery的原型对象中，添加插件的自定义函数，其中:

a. 为当前插件的父元素和子元素自动添加class

b. 为当前插件中的指定元素添加事件处理函数: 其实在原页面中已经实现了，所以只是一个提取的过程。无需重新编写。

6. 使用自定义插件: 用法和使用jquery-ui的插件步骤完全一样:

(1). 先编写插件的html

(2). 再引入插件的css和js

<link rel="stylesheet" href="插件.css">

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

<script src="插件.js">

(3). 在自定义的<script>中

$插件父元素.插件函数();

7. 示例: 封装jqueryui风格自定义插件——手风琴效果

my-ui/my-ui.css

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*accordion\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  .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;  }  .accordion>.content.fade{    height:0;    opacity:0;    overflow:hidden;    transition:all .5s linear;  }  .accordion>.content.in{    height:84px;    opacity:1;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*alert\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  .alert.in{    width:300px; height:100px;  } |

my-ui/my-ui.js

|  |
| --- |
| jQuery.fn.myAccordion=function(){    //1. 为插件中的元素自动添加class    //获得当前插件的父元素    var $parent=this;    //为插件父元素添加class accordion    $parent      .addClass("accordion")    //return $parent    //为父元素下的所有子元素中奇数位置的元素添加.title      .children(":nth-child(2n+1)") //记住      .addClass("title")    //return 所有奇数位置的元素: children(":nth-child(2n+1)")    //为每个奇数位置的下一个兄弟元素加class content和fade      .next() //自带遍历: 得到的是所有奇数位置的下一个兄弟元素，也就是所有偶数位置的元素      .addClass("content fade")    //return 所有偶数位置的元素    //为所有偶数位置的元素中第一个元素加class in      .first() //获得集合中第一个元素      .addClass("in");    //2. 为插件中的元素自动绑定事件处理函数    $parent.on("click",".title",e=>      $(e.target).next(".content").toggleClass("in")        .siblings(".content").removeClass("in")    );  }  //希望将来  //$("#my-accordion").myAccordion(); |

14\_accordion.html

|  |
| --- |
| <html>  <head>  <title> new document </title>  <meta charset="utf-8">  <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>  <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="my-ui/my-ui.js"></script>  <script>  //希望像jqueryui一样，用jquery对象调用一个函数就能应用插件效果  $("#my-accordion").myAccordion()  </script>  </body>  </html> |

三. ajax

$.ajax({

url:"服务器端接口地址", //服务器端接口地址

type:"get或post", //请求的类型

data:{ 参数: 值, ... : ... }, //要发送到服务器端的请求参数

dataType:"json", //让$.ajax()自动调用JSON.parse()

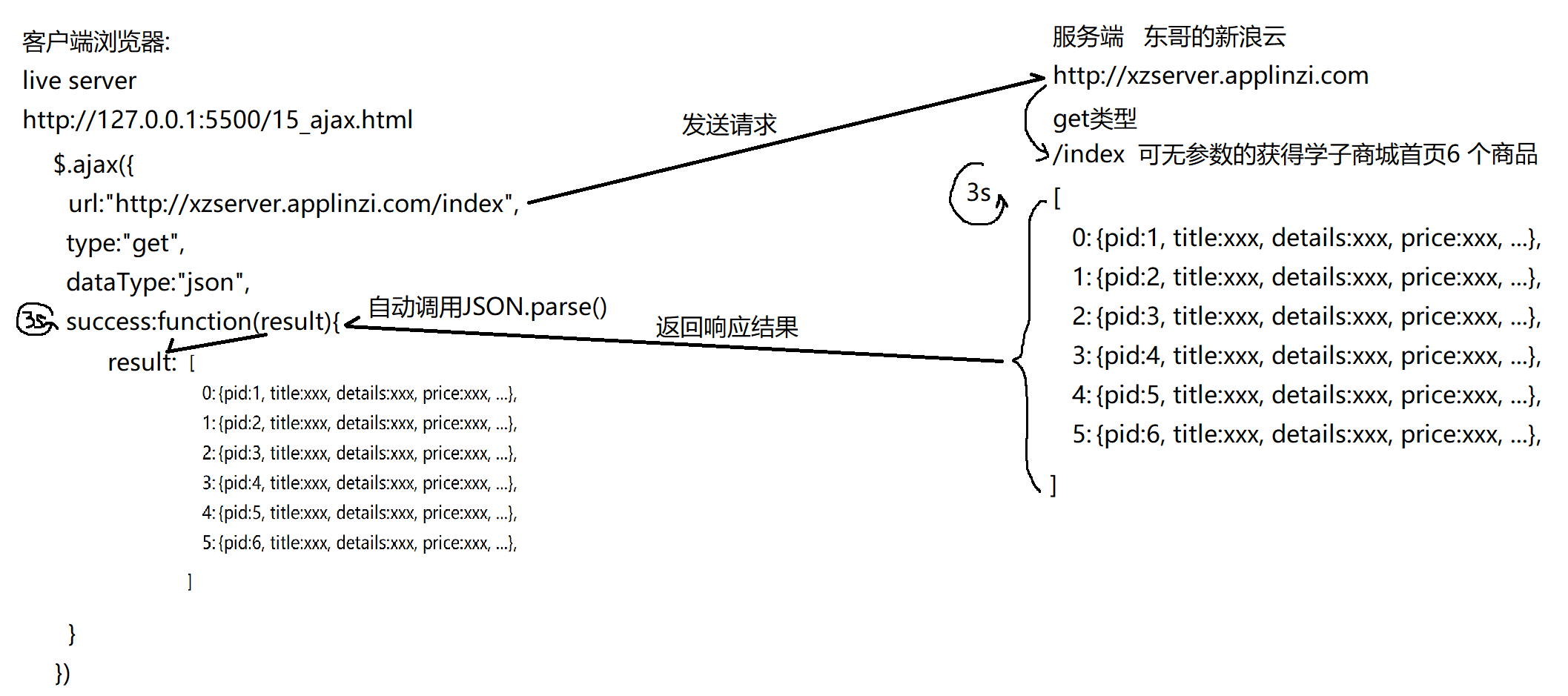
success:function(result){ //会在接收到服务器端响应结果后，自动执行该函数

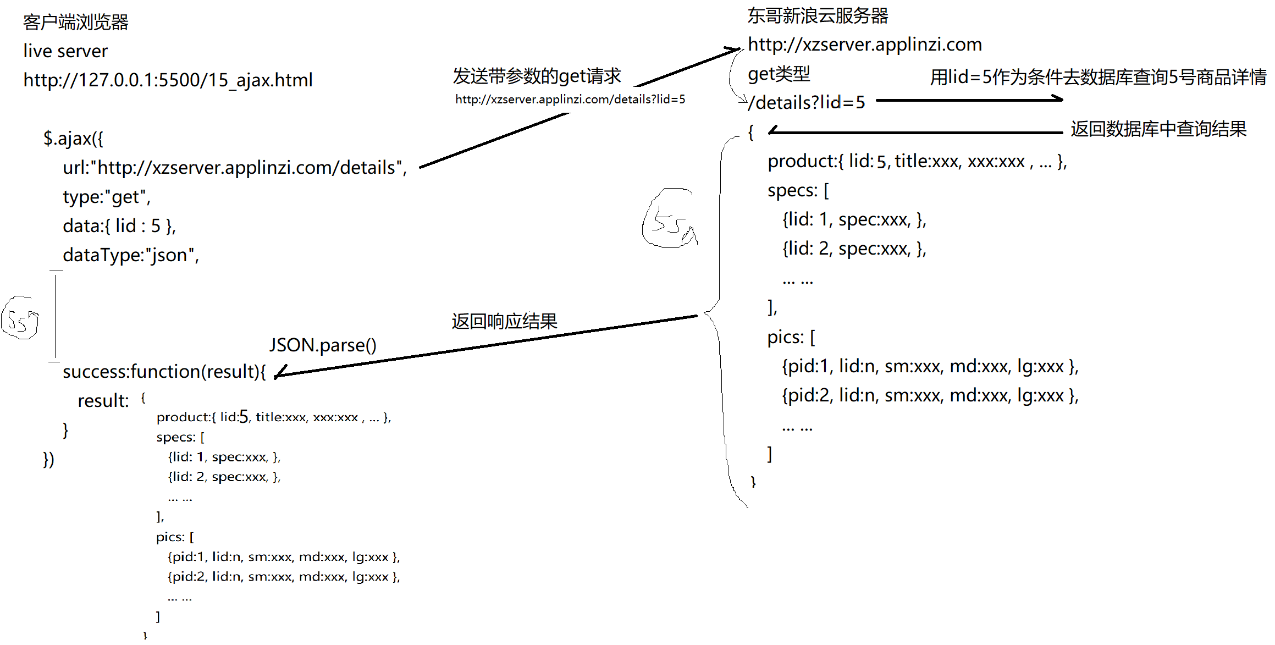
//result接住的就是服务器端返回的结果（已经被JSON.parse()自动转换了）

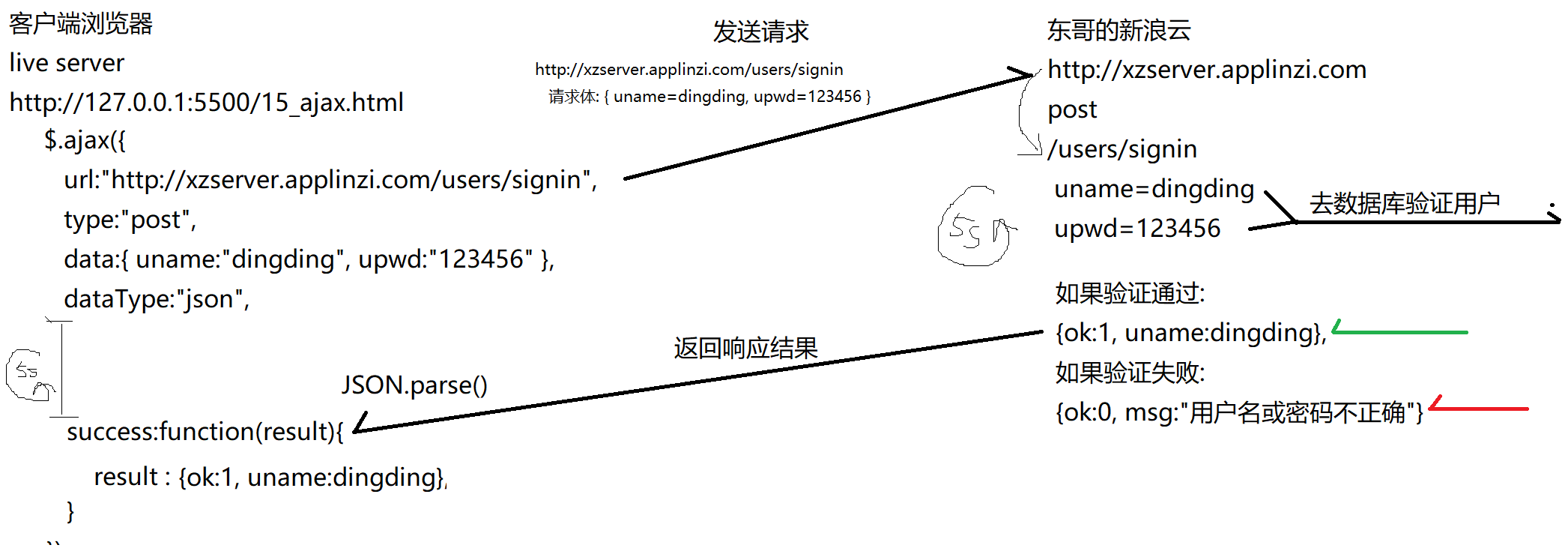
}

})

示例: 分别请求/index /details /users/signin接口







|  |
| --- |
| <body>  <script src="js/jquery-1.11.3.js"></script>  <script>    //请求学子商城首页的六个商品    $.ajax({      url:"http://xzserver.applinzi.com/index",      type:"get",      //data:{  },      dataType:"json",      success:function(result){        console.log(result);      }    })    //请求一个指定商品编号(5号)的商品的详细信息    $.ajax({      url:"http://xzserver.applinzi.com/details",      type:"get",      data:{ lid:5 },      dataType:"json",      success:function(result){        console.log(result);      }    })    //想请求登录接口验证用户名和密码    $.ajax({      url:"http://xzserver.applinzi.com/users/signin",      type:"post",      data:{uname:"dingding",upwd:"123456"},      dataType:"json",      success:function(result){        console.log(result);      }    })  </script>  </body> |

四. \*\*\*\*\*跨域:

1. 什么是跨域请求: 一个域名下的网站，向另一个域名下的服务器端请求资源(文件/接口)

2. 包括:

(1). 域名不同: http://www.a.com下的网页 向 http://www.b.com下的服务端发送请求

(2). 子级域名不同: http://oa.tedu.com下的网页 向 http://hr.tedu.com下的服务端发送请求

(3). 端口号不同: http://localhost:5500下的网页 向 http://localhost:5050下的服务器端发送请求

(4). 协议不同: http://12306.cn 下的网页 向 https://12306.cn下的服务端发送请求

(5). 即使是本机自己访问自己，域名下的网页向本机IP下的网页发送请求也算跨域:

http://localhost:5050 下的网页 向 http://127.0.0.1:5050 下的服务端发送请求

3. 问题: 单就浏览器而言

(1). 可以跨域:

<link rel="stylesheet" href="其他网站的css文件"> 比如CDN网络

<script src="其他网站的js文件"> 比如CDN网络

<img src="其他网站的图片">

<a href="跳转到其他网站的页面">

(2). 不能跨域: 唯独禁止ajax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(from)源头(origin)http://127.0.0.1:5500到(to)http://localhost:3000的xhr请求(Access)已经被(has been)CORS策略(policy)阻止了(blocked)。因为请求回来的资源(requested resource)没有(No)定义(present)" Access-Control-Allow-Origin "头部。

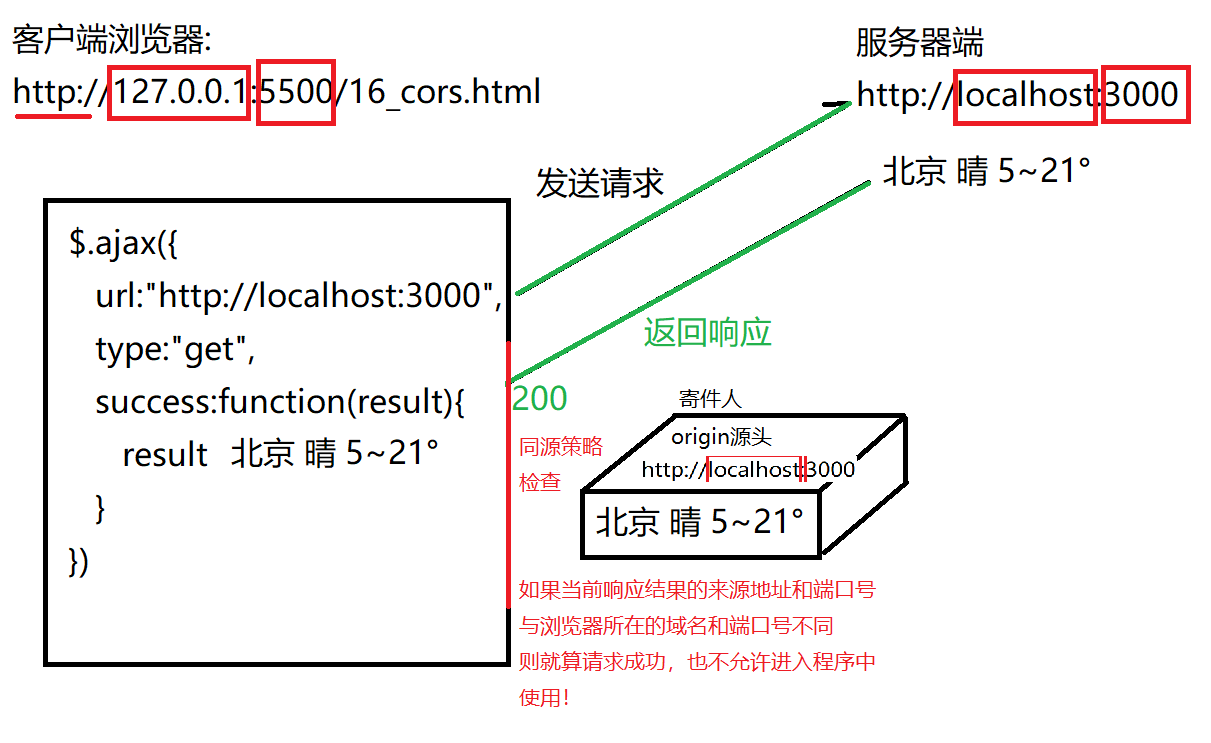
今后，只要看到这个错误！就是跨域错误！肯定域名、端口、协议至少有一个不一致！

4. 浏览器的同源策略(CORS policy):

(1). Cross Origin Resource Sharing

跨 不同源头 资源 共享

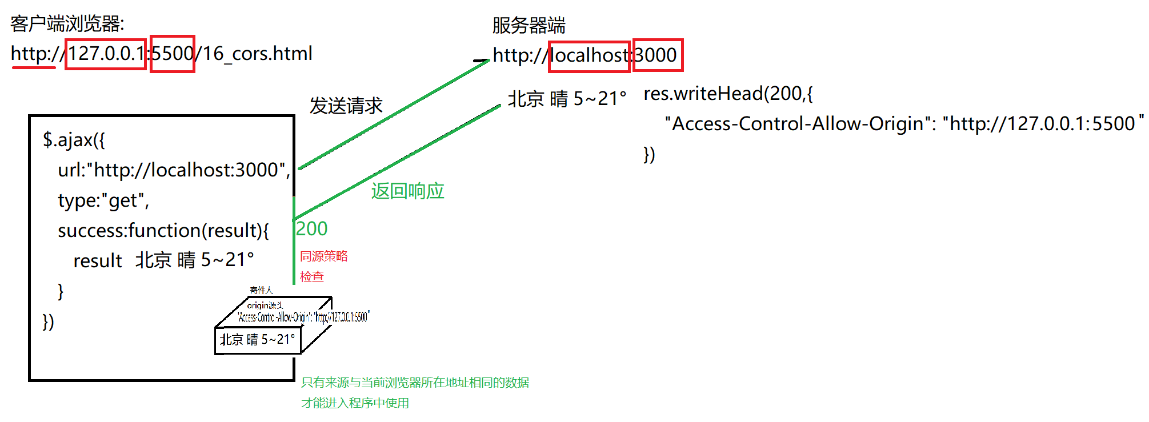
(2). 什么是: 浏览器禁止网页中的ajax使用与网页不同来源的服务端返回的数据。



5. 如何解决跨域问题: 2种方式:

(1). 服务器端CORS方式跨域:

a. 原理:



b. 示例: 使用服务器端cors方式跨域请求:

server.js

|  |
| --- |
| const http=require("http");  http.createServer((req,res)=>{    res.writeHead(200,{      "Content-Type":"text/plain;charset=utf-8",      "Access-Control-Allow-Origin":"http://127.0.0.1:5500"    })    res.write(`北京 晴 5~21°`);    res.end();  }).listen(3000);  //右键单击server.js，选择在终端/命令行中打开  //在弹出窗口中node server.js  //如果报端口错误，就关闭vscode，重新打开，重新运行node server.js  //打开浏览器，地址栏输入http://localhost:3000  //结果: 看到天气预报 |

16\_cors.html

|  |
| --- |
| <body>    <a href="http://localhost:3000">今日天气</a>  <script src="js/jquery-1.11.3.js"></script>  <script>    $.ajax({      url:"http://localhost:3000",      type:"get",      success:function(result){        console.log(result)//希望输出北京 晴 5~21°      }    });    //用live server打开网页  </script>  </body> |

c. 问题: CORS跨域只能被少数客户端地址使用。

1). 不好的解决: 可以把CORS后写死的一个地址，换成\*，表示可通过任意地址访问

问题: 一个服务器端，不可能随意允许任意客户端访问数据。

2). 好的解决: 如果一个服务器端真的需要让任意多个客户端访问，又需要有一定的安全性，则可以采用JSONP的方式。

(2). JSONP方式跨域: 看小程序视频！

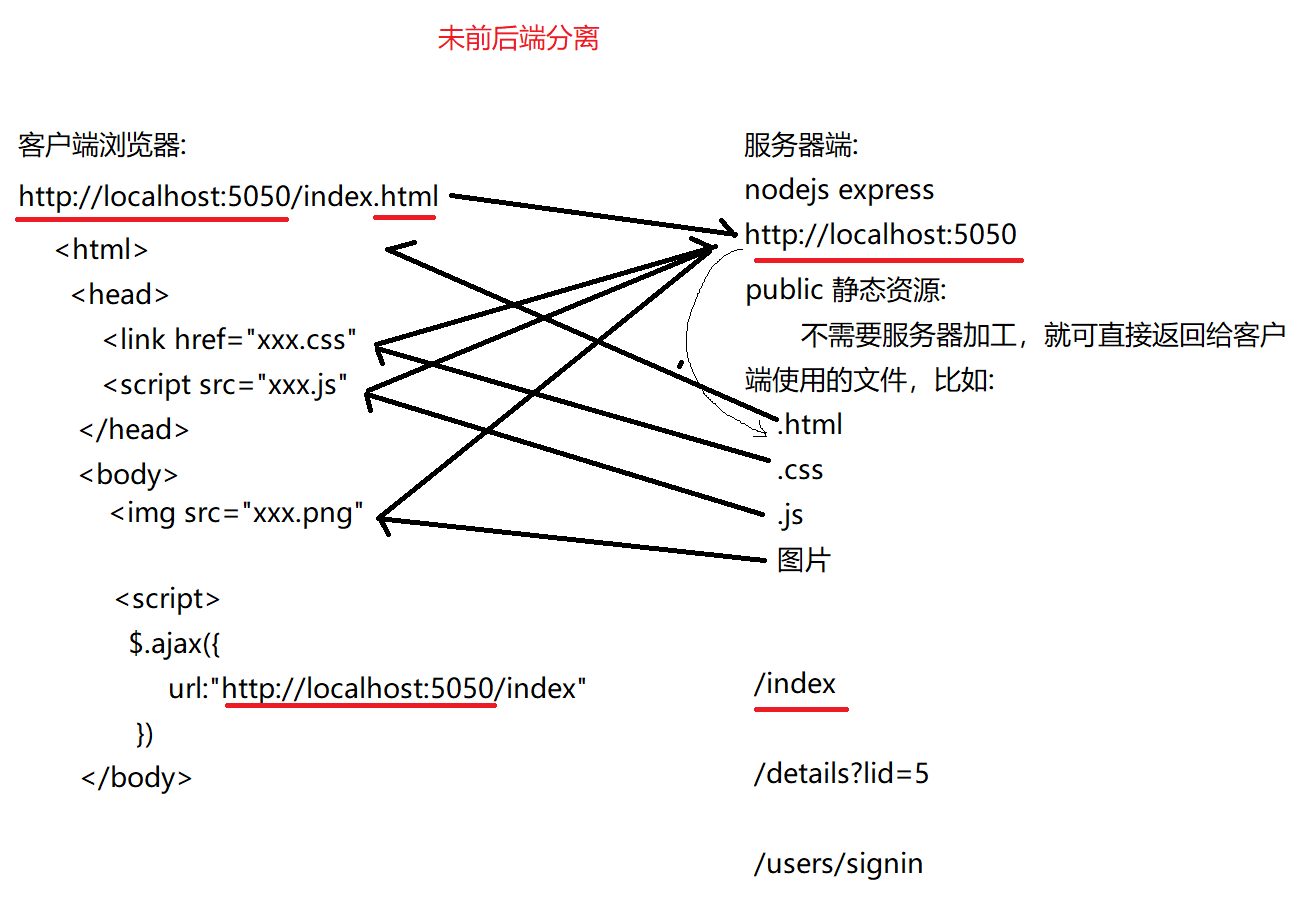
6. 前后端分离:

(1). 什么是: 今后一个大的项目，都是由前端和后端两个独立的项目组成的。前端项目和后端项目各自由不同的团队独立并行开发，甚至将来部署时，前端项目和后端项目都会运行在不同服务器的不同域名下。由不同的团队分别维护！

(2). 如果把前端项目和后端项目放在一个项目中不分离的缺点:

a. 紧耦合: 任何一方出错，另一方项目都会受牵连，无法正常运行！

b. 开发效率低: 前端很可能需要等待后端接口做完才能开始开发！串行！总项目时间会大大增加！



(3). 如果前后端分离，分别管理两个项目的优点:

a. 松耦合: 任何一方出错或延迟，另一方不会受牵连

b. 开发效率高: 今后都是前后端两个项目并行开发！在后端接口还没有做完的情况下，前端项目就已经可以开始了！

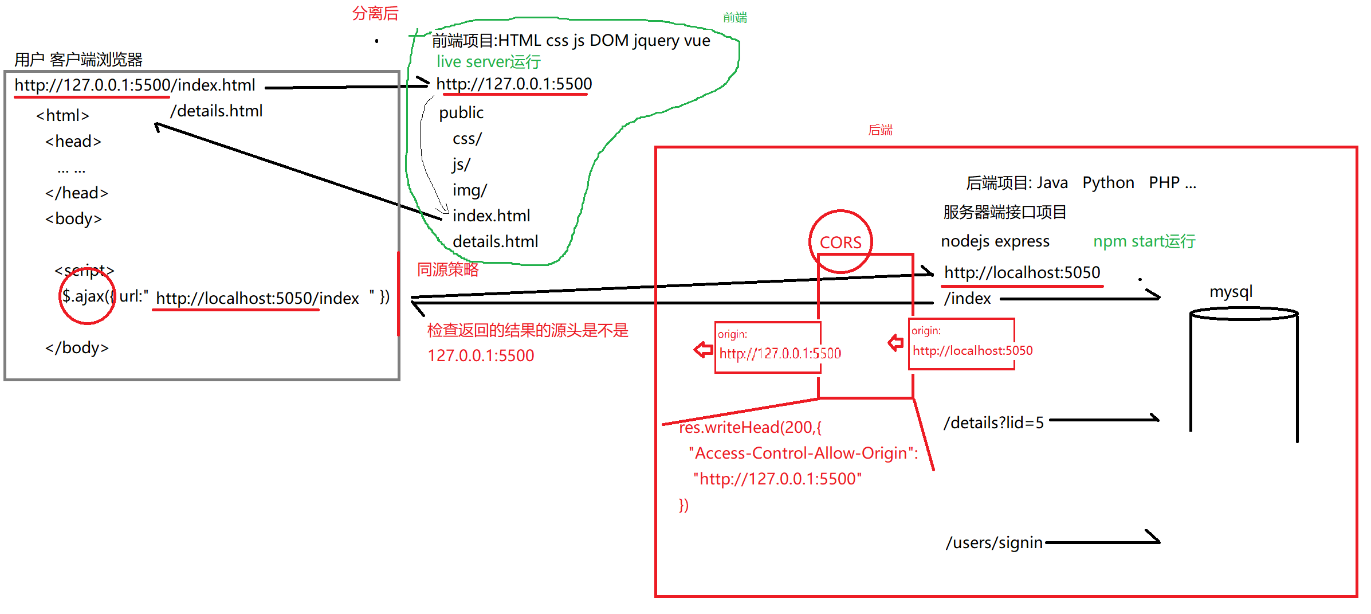
问题: 没有后端接口，怎么做前端页面的交互功能？

解决: 学会用最简单的方式模拟后端接口和数据，是每个前端必备的技能。

(1). 不嫌麻烦: 可以自己写nodejs express mysql

(2). 最优做法: 使用专业的接口模拟软件: json-server （看小程序视频）

(4). 前后端唯一的纽带，只有ajax请求和跨域



总结:

1. 查找元素:

(1). 不需要查找就可直接获得的元素: 5个

a. document jq中: $(document)

b. <html> document.documentElement jq中: $("html")

c. <head> document.head jq中: $("head")

d. <body> document.body jq中: $("body")

e. <form> document.forms[i] jq中: $("form")

(2). 按节点间关系查找: 2大类关系 6个属性

何时: 如果已经获得一个元素，想找周围附件的元素时

a. 父子关系: 4个属性

1). 获得一个元素的父元素: 元素.parentNode

Jq中: $元素.parent()

2). 获得一个元素下的所有直接子元素: 元素.children

Jq中: $元素.children("选择器")

在所有后代中查找符合条件的元素: $元素.find("选择器")

3). 获得一个元素下的第一个直接子元素: 元素.firstElementChild

Jq中: $元素.children(":first-child")

4). 获得一个元素下的最后一个直接子元素: 元素.lastElementChild

Jq中: $元素.children(":last-child")

b. 兄弟关系: 2个属性

1). 获得一个元素的前一个兄弟元素: 元素.previousElementSibling

Jq中: $元素.prev()

获得一个元素之前所有兄弟: $元素.prevAll("选择器")

2). 获得一个元素的后一个兄弟元素: 元素.nextElementSibling

jq中: $元素.next()

获得一个元素之后所有兄弟: $元素.nextAll("选择器")

3). 获得除当前元素之外所有兄弟元素: $元素.siblings("选择器")

(3). 按HTML特征查找: 4个方法

何时: 只按一个条件就可找到想要的元素时

a. 按id查找一个元素: var 一个元素=document.getElementById("id名")

var $jq对象=$("#id名")

b. 按标签名查找多个元素:

var 类数组对象=任意父元素.getElementsByTagName("标签名")

var $jq对象=$("元素选择器")

c. 按class名查找多个元素:

var 类数组对象=任意父元素.getElementsByClassName("class名")

var $jq对象=$(".class名")

d. 按name名查找多个表单元素

var 类数组对象=document.getElementsByName("name名")

var $jq对象=$("[name=name名]")

(4). 按选择器查找: 2个方法

何时: 如果查找条件复杂时

a. 只查找一个符合条件的元素:

var 一个元素=任意父元素.querySelector("选择器")

b. 查找多个符合条件的元素:

var 类数组对象=任意父元素.querySelectorAll("选择器")

c. jq中统一为: var $jq对象=$("任意css选择器")

2. 修改元素: 3样:

(1). 内容: 3种:

a. 获取或修改元素的原始HTML内容: 元素.innerHTML

jq中: $元素.html("新html内容")

b. 获取或修改元素的纯文本内容: 元素.textContent

jq中:$元素.text("新纯文本内容")

c. 获取或修改表单元素的值: 元素.value

jq中:$元素.val("新表单元素值")

(2). 属性: 3种:

a. 字符串类型的HTML标准属性: 2种

1). DOM中:

i. 核心DOM 4个方法:

元素.getAttribute()/setAttribute()/hasAttribute()/removeAttribute()

ii. HTML DOM: 元素.属性名

2). Jq中:

i. $元素.attr()统一了getAttribute()和setAttribute()

ii. $元素.prop()代替了 元素.属性名

b. bool类型的HTML标准属性:

1). DOM中:

i. 只能用HTML DOM: 元素.属性名 值必须是bool类型

~~ii. 不能用核心DOM4个方法~~

2). Jq中: 只能用$元素.prop()

c. 自定义扩展属性: 2种:

1). DOM中:

i. 核心DOM 4个方法:

元素.getAttribute()/setAttribute()/hasAttribute()/removeAttribute()

ii. 如果自定义属性名以"data-"为前缀，还可用: 元素.dataset.属性名

iii. ~~不能用HTML DOM~~

2). Jq中: 只能用 $元素.attr();

(3). 样式:

a. 只修改一个元素的一个css属性: 元素.style.css属性

jq中: $元素.css("css属性名","属性值")

b. 获取一个元素的css属性: var style=getComputedStyle(元素); style.css属性

jq中: $元素.css("css属性名")

c. 批量修改一个元素的多个css属性:

1). DOM中: 元素.className

2). Jq中: $元素.addClass()/removeClass()/hasClass()/toggleClass()

3. 添加元素:

(1). 只添加一个元素: 3步

a. 创建新的空元素对象: var新元素=document.createElement("元素名")

b. 为新元素添加关键属性: 新元素.属性名=值

jq中: 将前两步统一为了一句话

var $新元素=$(`HTML片段`)

c. 将新元素添加到DOM树: 3种:

1). 在父元素下所有子元素末尾追加一个新元素:

父元素.appendChild(新元素)

Jq中: $父元素.append($新元素)

$新元素. appendTo(父元素)

开头插入新元素: $父元素.prepend($新元素)

$新元素.prependTo(父元素)

2). 在父元素下的一个现有子元素前插入:

父元素.insertBefore(新元素, 现有子元素)

Jq中: $现有元素.before($新元素)

$新元素.insertBefore($现有元素)

在现有元素之后插入: $现有元素.after($新元素)

$新元素.insertAfter($现有元素)

3). 替换父元素下的一个现有子元素:

父元素.replaceChild(新元素, 现有子元素)

Jq中: $现有元素.replaceWith($新元素)

$新元素.replaceAll($现有元素)

(2). 添加多个元素时: 2种:

a. 同时添加父元素和多个子元素: 先将所有子元素加入到父元素中。最后再将父元素一次性添加到DOM树

b. 父元素已经在页面上，需要添加多个平级子元: 先将所有子元素加入到文档片段对象中，最后再一次性将文档片段对象加入到DOM树: 3步

1). 创建一个文档片段对象:

var frag=document.createDocumentFragment();

2). 将子元素加入到文档片段对象中:

frag.appendChild(新子元素)

3). 将文档片段对象整体加入DOM树

父元素.appendChild(frag)

c. jq中，都封装在了简化版函数内，很少再考虑这个问题了

4. 删除元素: 父元素.removeChild(子元素)

Jq中: $任意元素.remove()

5. 事件:

(1). 如何绑定事件处理函数: 2种:

a. 如果一个事件属性上只绑定一个处理函数: 赋值方式:

元素.on事件名=function(){ ... } 不推荐使用

b. 如果一个事件属性上想同时绑定多个处理函数: 添加事件监听对象:

元素.addEventListener("事件名", 事件处理函数)

Jq中:

标准写法: $元素.on("事件名", 事件处理函数)

简写: 如果所用事件在以下常用事件列表中，则可简写为:

$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 |

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

Jq中: $元素.off("事件名",原事件处理函数对象)

(2). 事件模型: 3个阶段: 捕获、目标触发、冒泡执行

(3). 事件对象:

a. 如何获得事件对象:

1). DOM中: 元素.on事件名=function(e){ ... }

2). Jq中:$元素.on("事件名", function(e){ ... })

b. 取消冒泡: e.stopPropagation();

c. 利用冒泡/事件委托:

1). DOM标准做法:

i. 优化: 如果多个平级子元素都要绑定相同的事件处理函数时，应该只在父元素上绑定一次处理函数即可！——结果: 点击任何一个子元素，都会冒泡到父元素上执行

ii. 问题: 如果事件绑定在父元素上，this->父元素，不再指实际出发事件的子元素。

解决: 只要使用事件委托，都要用e.target代替this，获得目标元素

iii. 问题: 如果只希望子元素中部分想要的元素执行事件处理函数，而不希望所有子元素都执行事件处理函数.

解决: 只要使用事件委托，都要先判断e.target是否想要的。

判断依据可以是任何条件: nodeName, className, innerHTML...

2). Jq中:

i. 事件还是绑定在父元素上一份即可:

ii. 不用自己写if判断目标元素，而是为on添加第二个实参值: 选择器条件

iii. this重新指向e.target

$父元素.on("事件名","选择器条件",function(){

$(this) 代替 $(e.target)

})

d. 取消默认行为: e.preventDefault()

e. 获得鼠标位置坐标: 3组坐标:

1). 鼠标所在位置相对于显示屏左上角的距离: e.screenX e.screenY

2). 鼠标所在位置相对于浏览器内文档显示区左上角的距离:

e.clientX e.clientY

3). 鼠标所在位置相对于事件绑定的元素左上角的距离:

e.offsetX e.offsetY

f. 页面滚动事件:

1). 在页面滚动时，自动执行:

// 当滚动时自动执行

window.onscroll=function(){

//希望获得页面滚动过的距离: body顶部超出文档显示区顶部的距离

var scrollTop=document.documentElement.scrollTop//能用第一就用第一个

||document.body.scrollTop//如果不能用第一个就退一步用第二个

}

2). 用程序主动控制让网页滚动到哪个位置:

window.scrollTo(水平方向滚动到的位置, 垂直方向滚动到的位置)

// 滚动 0

(4). 页面加载后自动执行: 2次页面加载完成事件

a. 仅HTML内容和js加载完成就提前触发: (不用等CSS和图片)

DOMContentLoaded

Jq中: 完整: $(document).ready(function(){ ... })

最简化: $(()=>{ ... })

b. 所有页面内容都加载完成: (必须等待HTML+CSS+JS+图片)

DOM中: window.addEventListener("load",function(){ ... })

Jq中: $(window).load(function(){ ... })

(5). 鼠标事件:

a. DOM中: mouseover mouseout

问题: 反复进出子元素，也会反复触发父元素上的进出事件，产生歧义，与现实不符

b. jq中: mouseenter mouseleave

优点: 即使反复进出子元素，也不会反复触发父元素上的进出事件。不会产生歧义，也与现实相符。

简写: 如果同时绑定mouseenter和mouseleave时，应该只绑定hover即可

$元素.hover( //mouseenter + mouseleave

function(){ ... }, //mouseenter

function(){ ... } //mouseleave

)

(6). 模拟触发:

a. 标准写法: $元素.trigger("事件名")

b. 如果事件属于常用事件列表中: $元素.事件名()

总结: jq中常用的好用的函数:

1. 判断一个元素是否具有什么特征: $元素.is("选择器")

2. 遍历jquery查找结果中每个DOM元素: $元素.each(function(i, elem){ ... })

3. 查找一个元素在查找结果中的位置: var i=$查找结果.index(要找的元素)

总结: $的用法: 4种

1. $("选择器") 查找DOM元素并包装进jQuery对象中

2. $(DOM元素) 无需查找，直接将一个DOM元素包装进jQuery中，$(this)

3. $(`HTML片段`) 创建新元素

4. $(function(){ ... }) 绑定DOM内容加载完成(DOMContentLoaded)就提前触发的事件处理函数

总结: jQuery简化版函数三大特点:

1. 自带for循环/遍历: 对整个jQuery对象调用一次简化版函数，等效于对jQuery对象中的每个DOM元素都调用一次等效的原生DOM函数或属性.

2. 一个函数两用: 凡是和获取/修改有关的函数，如果调用函数时没给新值，默认就读取旧值。如果给了新值，则自动切换为修改元素的属性值或内容。

3. 绝大多数简化版函数都会返回正在操作的.前的jQuery对象: 如果后续操作，依然是作用于同一个jquery对象上，可以继续连着写！——链式操作！

总结: this

1. obj.fun() this->obj

2. fun() 或 (function(){ ... })() 或 回调函数 this->window

3. new Fun() this->new正在创建的这个子对象

4. 构造函数.prototype.fun=function(){ ... } this->将来调用这个方法的某个子元素

5. 访问器属性中的this->当前访问器属性所在对象

Object.defineProperties(eric,{

eage:{

get(){return this.\_eage},

set(value){ this.\_eage=value }

}

})

6. 元素.onclick=function(){ ... } this->当前触发事件的DOM元素

7. $元素.click(function(){ ... }) this->当前触发事件的DOM元素

.addEventListener()

作业:

1. (必须)看小程序视频学习jsonp跨域

小程序->在线->JQUERY->day04 作业: 高频笔试题jsonp跨域

2. (必须)看小程序视频继续完成学子商城jquery版项目:

强调: 视频中所有http://localhost:5050/xxx服务端接口地址，一律改为: http://xzserver.applinzi.com/xxx

小程序->在线->JQUERY->day04

项目: 1. 学子商城项目——首页 index jquery ajax 请求 填充页面内容

项目: 2. 学子商城项目——页头 header jquery ajax加载公共的页头

项目: 3. 学子商城项目——详情页 details 页面间传值

3. (扩展)看小程序视频继续完成学子商城jquery版项目:

强调: 视频中所有http://localhost:5050/xxx服务端接口地址，一律改为: http://xzserver.applinzi.com/xxx

小程序->在线->JQUERY->day04

项目: 4. 学子商城项目——放大镜效果 zoom 鼠标位置 背景定位

4. (扩展)看小程序源代码学习炫酷广告轮播特效:

小程序->在线->JQUERY->day04 婚纱网站 轮播切换图片特效 单击切换图片 ...