20170225课堂笔记

AJAX

> 作用：
>
> - 如果客户端需要获取的是资源文件，浏览器会自动帮我们向服务器端发送请求，并且接受服务器返回的内容进行渲染
>
> - 在地址栏输入网址
> - 利用LINK
> - 利用SCRIPT
> - 利用IFRAME
> - …
> - 但是如果我们需要请求的是数据，就需要使用AJAX等技术发送请求了(AJAX是JS中的一个核心重要知识点)
>
> 什么是AJAX：
> Async(Asynchronous) JavaScript And Xml 异步的JS和XML
>
> - 什么是XML
> - AJAX叫做异步的,难道使用异步请求吗?

扫盲1：什么是XML

html：超文本标记语言，W3C制定了很多具有语义化的标记标签，我们使用这些HTML标签搭建页面结构(v4/v5)
xhtml：更加严谨严格的HTML
dhtml：页面中的数据是动态绑定的
xml：可扩展的标记标签语言，它里面使用的标签都是自己定义的，不是W3C的规范标签 ->我们可以使用自己定义的有意义的标签来存储数据,这样结构清晰明了(目前前端数据存储和展示一般都使用JSON,但是之前一般都是XML)
wxml：微信小程序的页面就是.wxml，小程序中使用的标签都是小程序自己定义的(微信XML)

> 在以前的项目开发中，服务器端返回给客户端的数据类型一般都是XML格式的，只不过XML格式的数据在客户端的二次解析过程中相对于JSON比较麻烦，现在服务器端返回给客户端的数据一般都是JSON格式的数据；但是也有XML格式的数据；

扫盲2：异步的JS->异步刷新

这里的异步不能理解为前面讲的JS中的同步异步编程，这里面的异步指的是“局部刷新”，如果用七个字概述AJAX的作用，那就是实现局部刷新的

web1.0或者web2.0初期 =>整体刷新
我们前端页面中的功能和数据绑定大部分都是后台开发者使用后台语言来完成的；浏览器只有一个作用，就是把后台实现好功能的页面呈现即可，所有的操作都是后台在服务器完成的；这样的话如果前端页面中的内容需要改变的话，必须后台重新的把最新的内容返回，前端需要重新呈现，导致前端页面刷新

AJAX的基础知识

1、创建AJAX对象

```
1.  var xhr=new XMLHttpRequest(); //->在IE6及更低版本浏览器下不兼容，不兼容的话使用 new ActiveXObject 来完成
```

2、打开一个请求的URL地址

```
xhr.open([method],[request url],[async/sync],[username],[userpass])

/*
 * request url：请求数据的接口地址，客户端也是通过这个地址向服务器发送请求的
 * 客户端可以通过问号传递参数的方式把内容传递给服务器，例如：xhr.open('get','temp.json?name=zhufeng&age=8',true)
*/

/*
 * async/sync：同步或者异步，默认是TRUE代表异步，写FALSE是同步，不写就是TRUE；在真实项目中，为了防止出现请求堵塞的问题，我们大部分采用的是异步
*/

/*
 * method：请求方式
 * GET系列
 *    - get：获取,应用于从服务器获取内容
 *    - delete：删除，应用于删除服务器上的内容
 *    - head：头，应用于只想获取服务器响应头信息
 *
 * POST系列
 *    - post：推送，应用于给服务器推送内容
 *    - put：放，应用于在服务器上放文件等
 *
 * 还有更多的请求方式
 *
 * 值得注意的是：以上的几种方式，客户端都可以把内容传递给服务器，服务器也否可以把内容返回给客户端，只不过在不同的场景，我们使用对应的类型会更好一些，例如：我们给服务器的少,从服务器拿的多，我们最好使用GET请求，如果
```

```
    是给服务器的多，从服务器获取的少，我们最好使用POST请求；
26. */
27. //----------------------
28. /*
29.  * GET系列请求和POST系列请求的区别?
30.  * 核心：
31.  * - GET请求传递给服务器的内容是通过问号传参的方式传递的；
32.  *    xhr.open('get','temp.json?xxx=xxx&xxx=xxx')
33.  *
34.  * - 而POST传递给服务器的内容是通过请求主体传递的；
35.  *    xhr.send(...)
36.  *
37.  * 1、GET请求传递给服务器的内容没有POST请求多
38.  *    每一个浏览器对于URL的长度都有限制(谷歌8KB 火狐7KB IE2KB)，如果GET请求下传递的比较多，URL就会超出限制，超出限制的部分浏览器自动截取，导致服务器获取的内容不完整
39.  *    请求主体中的大小理论上没有限制的，但是真实项目中为了保证传输的速度，我们会限制传递内容的大小
40.  *
41.  * 2、GET请求容易出现缓存
42.  *    还是因为使用问号传参的方式，如果重复向同一个地址发送请求，传递的参数值都是一样的，浏览器会默认的给做缓存(这个缓存不可控)，所以我们一般项目中在使用GET请求的时候都要把缓存清除掉
43.  *    xhr.open('get','temp.json?_='+Math.random())
44.  * 为什么属性名使用_,我换成其他的行不行? 因为除了传递随机数以外，我们可能还会传递别的，这样的话如果随机数的名字和其他的相同就冲突了，用_杜绝冲突。
45.  *
     * 3、GET请求相对于POST来说不安全
```

```
46.     *      有一种黑客技术叫做“URL劫持”，被劫持后，问号后面
        的参数值都会被获取或者修改，导致不安全
47.     */
```

3、监听AJAX状态改变，在不同状态下处理不同事情

```
xhr.onreadystatechange=function(){
    if(xhr.readyState===4 && xhr.status===200){
        var val=xhr.responseText;
        //->xhr.getResponseHeader([key]) 获取响应头信息
        //->responseText：获取的是字符串(一般都是服务器响应主体返回的JSON字符串)
        //->responseXML：获取的是XML数据
    }
}
//->面试题：在你以前的项目中服务器返回的数据一般都是什么格式的?
//1、返回的是字符串格式的
// + 普通的字符串
// + JSON格式的字符串(最常用的)
// + 二进制或者文件流编码格式的字符串(一般请求的是图片,服务器返回的都是二进制编码字符串)
//2、返回的是XML文档格式的

/*
 * xhr.readyState：AJAX状态码
 * 0 UNSENT 未发送,刚开始创建完成AJAX对象,默认的状态就是0
 * 1 OPENED 已打开,执行了xhr.open之后状态变为1
 * 2 HEADERS_RECEIVED 响应头信息已经成功的返回并且被接收
 * 3 LOADING 响应主体内容正在加载
 * 4 DONE 响应主体内容接收成功
*/

/*
 * xhr.status：服务器返回的HTTP网络状态码
 * 200 请求成功
```

```
 *
 * [成功，只是中间经历了转折]
 * 301 永久重定向(永久转移) 域名更换的时候我们基本上都会用301做永久的重定向
 * 302 临时重定向(临时转移) ->307(临时重定向)  服务器负载均衡,例如：一台服务器最高并发数在500左右，当第501个人过来的时候，当前服务器不能有效的进行处理，此时我们需要把此客户端的请求临时转移到另外的一台服务器上进行处理
 * 304 读取的是缓存的数据 在真实的项目中,产品一但上线,资源图片、JS、CSS等内容是不轻易改变的，此时我们最好做一下304缓存：第一次向服务器发送请求来访问的时候，把加载完成的资源文件进行缓存，第二次直接读取缓存中的数据即可，减少服务器压力
 *
 * [客户端的错误]
 * 400 请求参数有误
 * 401 请求的权限不够
 * 404 请求的地址不存在
 *
 * [服务器端错误]
 * 500 未知的服务器端错误
 * 503 服务器超负荷
*/
```

4、发送AJAX请求给服务器

```
xhr.send(null);
//->GET系列的请求传递的一般都是null，因为他们通过问号传参把内容传递给服务器
//->POST系列的请求会把需要传递给服务器的内容放在这里(请求主体)
```

AJAX这件事(这个任务)
开始：xhr.send(…)
结束：xhr.readyState===4

AJAX实战

面试题：你之前的项目中是否做过倒计时之类的东西吗？你们做的时候当前时间是从客户端本地读取的还是从服务器读取的？从服务读取时间你是怎么解决时间差的？

```
/*
 * 获取服务器的时间,我们不需要再响应主体中获取,在响应头的信息中,有一个叫做Date的属性,它存储的值就是服务器的时间
 */
var xhr = new XMLHttpRequest();
xhr.open('head', 'temp.xml?_=' + Math.random());//->为什么选择HEAD请求方式?因为当前的需求,我们只想获取到服务器时间,这样的话只需要把响应头信息获取到即可,主体内容不需要获取了,使用HEAD就是只获取头信息,加大请求的效率
xhr.onreadystatechange = function () {
    if (xhr.status !== 200) return;
    if (xhr.readyState === 2) {//->我们只需要响应头信息返回就可以获取到服务器的时间,如果等到4的时候,虽然也可以获取到,但是间隔的时间更长了,导致时间差也会变大(真实时间和服务器获取的时间差值)
        var time = xhr.getResponseHeader('Date');//->获取到的时间是格林尼治时间(GMT),我们还需要把这个时间变为北京时间(GMT+0800)
        time = new Date(time);
        console.log(time);
    }
};
xhr.send(null);
```