Ajax

Day 01

**AJAX技术**

即 Asynchronous Javascript And XML，AJAX 不是一门的新的语言，而是对现有持术的综合利用。

本质:是在HTTP协议的基础上通过js的XMLHttpRequest对象与服务器进行通信。

作用：可以在页面不刷新的情况下，请求服务器，局部更新页面的数据；

### **异步 与 同步(了解)**

指某段程序执行时不会阻塞其它程序执行，其表现形式为程序的执行顺序不依赖程序本身的书写顺序，相反则为同步。

同步：同一时刻只能做一件事，上一步完成才能开始下一步

异步：同时做多件事，效率更高,做一件事情时，不影响另一件事情的进行。

**XMLHttpRequest可以以异步方式的处理程序。**

### **发送get请求**

XMLHttpRequest以异步的方式发送HTTP请求，因此在发送请求时，一样需要遵循HTTP协议。

//使用XMLHttpRequest发送get请求的步骤

1. 创建一个XMLHttpRequest对象

var xhr = new XMLHttpRequest();

2. 设置请求行

第一个参数:请求方式 get/post

第二个参数:请求的地址 需要在url后面拼上参数列表

xhr.open("get", "08.php?name=hucc");

3. 设置请求头

xhr.setReqeustHeader('content-type','text/html');

//浏览器会给我们默认添加基本的请求头,get请求时无需设置

4. 设置请求体

//get请求的请求体为空,因为参数列表拼接到url后面了

xhr.send(null);

get请求,设置请求行时,需要把参数列表拼接到url后面

get请求不用设置请求头

get请求的请求体为nul

### **发送post请求**

* var xhr = new XMLHttpRequest;
* //1. 设置请求行 post请求的参数列表在请求体中
* xhr.open("post", "09.php");
* //2. 设置请求头, post请求必须设置content-type,不然后端无法获取到数据
* xhr.setRequestHeader("content-type", "application/x-www-form-urlencoded");
* //3. 设置请求体
* xhr.send("name=hucc&age=18");

post必须设置请求头中的content-type为application/x-www-form-urlencoded

post请求需要将参数列表设置到请求体中

**获取响应**

HTTP响应分为3个部分，状态行、响应头、响应体。

//给xhr注册一个onreadystatechange事件，当xhr的状态发生状态发生改变时，会触发这个事件。 //3-监听服务器响应

xhr.onreadystatechange = function () {

if (xhr.readyState == 4 && xhr.status ==200) {

var r = xhr.responseText; //获取服务器返回数据

console.log(r);

//先将返回内容进行包装，

//将聊天内容显示在聊天窗口中

messages.appendChild( setMsg('other', r));

}

}

**readyState ：**记录了XMLHttpRequest对象的当前状态

//0：请求未初始化（还没有调用 open()）。

//1：请求已经建立，但是还没有发送（还没有调用 send()）。

//2：请求已发送，正在处理中

//3：请求在处理中；通常响应中已有部分数据可用了，但是服务器还没有完成响应的生成。

//4：响应已完成；您可以获取并使用服务器的响应了。

**XML**

**什么是XML**

1.XML 指可扩展标记语言（EXtensible Markup Language）

2.XML 是一种标记语言，很类似 HTML

3.XML 的设计宗旨是传输数据，而非显示数据

4.XML 标签没有被预定义。您需要自行定义标签。

5.html与XML 的 区别：html用于展示数据XML用于存储和传递数据

**语法规范**

1.第一行必须是版本信息

2.必须有一个根元素（有且仅有一个）

3.标签不可有空格、不可以数字或.开头、大小写敏感

4.不可交叉嵌套，都是双标签，如果是单标签，必须闭合

5.属性双引号（浏览器自动修正成双引号了）

6.特殊符号要使用实体

7.注释和HTML一样

**php获取xml文件的内容**

**//注意，如果需要返回xml数据，需要把content-type改成text/xml,不然浏览器以text/html进行解析。**

**header('content-type:text/xml;charset=utf-8');**

**//用于获取文件的内容**

**//参数：文件的路径**

**$result = file\_get\_contents("data.xml");**

**echo $result;**

### **JSON数据**

JSON(JavaScript Object Notation, JS 对象标记) 是一种轻量级的数据交换格式。它基于 ECMAScript 规范的一个子集，采用完全独立于编程语言的文本格式来存储和表示数据。

1.数据在名称/值对中

2.数据由逗号分隔(最后一个健/值对不能带逗号)

3.花括号保存对象，方括号保存数组

4.键使用双引号

var obj = {a: 'Hello', b: 'World'}; //这是一个对象，注意键名也是可以使用引号包裹的

var json = '{"a": "Hello", "b": "World"}'; //这是一个 JSON 字符串，本质是一个字符串

**JSON数据在不同语言进行传输时，类型为字符串，不同的语言各自也都对应有解析方法，需要解析完成后才能读取**

#### **php处理json**

php关联数组==> json $json = json\_encode($obj)

json===>php对象 $obj = json\_decode($json,true);

JS对象 ==> JSON字符串 JSON.stringify(obj)

JSON字符串 ==> JS对象 JSON.parse(obj)

**Ajax**

var $={

ajax:function(obj){

//获取用户的参数

var type=obj.type||'get'; //默认请求方式是get

var url=obj.url||location.href; //默认请求当前页面

var callback=obj.callback;

//1-js中使用对最方便接受的参数是对象，但是传递给服务器的格式 name=zs&age=18

var data=this.setParam(obj.data); //name=zs&age=18

// console.log(data);

//封装ajax公共代码部分

//1-创建XMLHttpRequest对象

var xhr=new XMLHttpRequest();

//模拟http协议

//如果是get请求在url后面拼接参数

if(type=='get'){

url=url+'?'+data;

data=null;

}

//1-请求行

xhr.open(type,url);

//2-请求头 post 必须设置请求头

if(type=='post'){

xhr.setRequestHeader('content-type','application/x-www-form-urlencoded');

}

//3-请求主体

xhr.send(data);

//监听服务器的响应

xhr.onreadystatechange=function(){

if(xhr.readyState==4 &&xhr.status==200){

var r=xhr.responseText;//获取响应主体

//r中就是服务器返回核心数据 需要渲染

callback&&callback(r);

}

}

},

//将对象转成 name=zs&age=18

setParam:function(obj){

if(typeof obj =='object'){

var str='';

for(var k in obj){

str+=k+'='+obj[k]+'&';

}

str=str.substr(0,str.length-1); //参数一：开始索引 参数二：截取长度

}

return str;//返回转换后的字符串

}

};

**$.ajax**

$.ajax({

type:"get",//请求类型

url:"02.php",//请求地址

data:{name:"zs", age:18},//请求数据

dataType:"json",//希望接受的数据类型

timeout:5000,//设置超时时间

beforeSend:function () {

//alert("发送前调用");

},

success:function (data) {

//alert("成功时调用");

console.log(data);

},

error:function (error) {

//alert("失败时调用");

console.log(error);

},

complete:function () {

//alert("请求完成时调用");

}

});

**表单序列化**

**serialize()方法序列化表单**

就是将表单中带有name属性的所有参数拼成一个格式为name=value&name1=value1这样的字符串。方便我们获取表单的数据。

$('form').serialize();

**模板引擎**

## **为什么要使用模板引擎**

我们通过ajax获取到数据后，需要把数据渲染到页面，在学习模板引擎前，我们的做法是大量的拼接字符串，对于结构简单的页面，这么做还行，但是如果页面结构很复杂，使用拼串的话**代码可阅读性非常的差，而且非常容易出错，后期代码维护也是相当的麻烦。**

### **artTemplate入门**

1. **引入模板引擎的js文件**
2. **准备模板**
3. **准备数据**
4. **将模板与数据进行绑定**
5. **将数据显示到页面**

**注意：传递给模板引擎的数据必须是对象**

**//如果返回的数据是个数组，必须使用对象进行包裹，因为在{{}}中只写书写对象的属性。**

### **同源策略的基本概念**

**协议相同 域名相同 端口相同**

同源政策的目的，是为了保证用户信息的安全，防止恶意的网站窃取数据。

限制： //1. Cookie、LocalStorage 无法读取。

//2. DOM 无法获得。

//3. AJAX 请求不能发送。

## **跨域**

**跨域行为是浏览器行为，是浏览器阻止了ajax行为。服务器与服务器之间是不存在跨域的问题的**

## **1.Jsonp**

使用起来相当的简单，跟普通的get请求没有任何的区别，只需要把dataType固定成jsonp即可。 dataType:"jsonp",

2.如果返回的响应头中设置了header('Access-Control-Allow-Origin:请求源域名或者\*')

# **XMLHttpRequest 2.0**

XMLHttpRequest是一个javascript内置对象，使得Javascript可以进行异步的HTTP通信

老版本的XMLHttpRequest的缺点：

1. 仅支持传输文本数据，无法传说二进制文件，比如图片视频等。

//2. 传输数据时，没有进度信息，只能提示完成与否。

//3. 受到了"同源策略"的限制

新版本的功能：

1. 可以设置timeout 超时时间

//2. 可以使用formData对象管理表单数据

//3. 可以获取数据传输的进度信息

## **formData管理表单数据**

1. 实例化一个formData对象， new formData(form); form就是表单元素

2.. formData对象可以直接作为 xhr.send(formData)的参数。注意此时数据是以二进制的形式进行传输。

3. formData有一个append方法，可以添加参数。formData.append("id", "1111");

4. 这种方式**只能以post形式传递**，**不需要设置请求头**，浏览器会自动为我们设置一个合适的请求头。

## **显示文件进度信息（progress）**

1. 需要注册 **xhr.upload.onprogress = function(e){}** 事件，用于监听文件上传的进度.注意：需要在send之前注册。

2. 上传的进度信息会存储事件对象e中

3. e.loaded表示已上传的大小 e.total表示整个文件的大小