**AJAX(**异步的js 和XML**)**

**问：**异步可以使用的发送的请求方式?

1. 在浏览器地址栏直接输入
2. 点击超链接
3. 提交表单

4. 通过AJAX可以在浏览器由向服务器发送异步请求

AJAX**同步请求：**

* 发送同步请求时，在页面加载时完成前 我们不能做任何的操作，只能等待页面加载z
* 这样当网速过慢的时，会带来一个非常不好的用户体验
* 同步请求一旦发送就要刷新全部的界面---就像那个正则验证

AJAX**异步请求：**

* 发送异步请求时，不会影响页面的其他部分，会自动浏览器的后台发送请求，
* 当响应数据返回时，在通过DOM将数据加载页面中，这样发送求不会影响用户的其他操作
* 会给用户带来一个非常好的体验

缺点：

通过AJAX加载的内容，不能使用回退按键回退

通过AJAX加载的内容，不利于搜索引擎的的检索，不利于SEO

AJAX无法发送跨域请求，通过AJAX的请求必须符合同源策略

同源策略是指协议名 域名/ip地址，端口号也必须完全一致，不符合同源策略的请求陈伟跨域请求

**XML(**可扩展的标记语言**)**

他的作用和HTML类似，不同的是HTML中都是预定义标签

XML中没有预定义标签，全部都是自定义标签，有且只有一个根标签用来表示一些数据文件较大，传输性差，可读性较好，用于做配置文件

**区别：**

**html：**<name=’wukong’,age:18>

**XMl：**<stuend><name>wokong<name><age>18</age><studend>

**json:** {“name”:”wukong”,”age”:18}

* 所以现在都用json传数据，XML淘汰，淘汰

**AJAX的使用：**

核心对象是：XMLHttpRequest

AJAX的所有操作都是通过该对象进行的

**使用步骤：**

1. **创建XMLHttpRequest对象**

var xhr = new XMLHttpRequest();

1. **设置请求信息**

xhr.open(method , url);(“是请求的方式：get/post”,路径)

**get请求的方法**

直接发在那边接受响应回来，可以在open里面传参数

**post请求的方法：**

1.首先发送的时候，要设置请求头：post请求必须设置如下的请求头，这样请求体才会被解析  
xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");  
2.发送的参数要写在send（）里面，

3.在rounder中要用body来接收，并且检查有没有中间件是否写上去了

1. **发送请求**

send()用来将请求发送给服务器，它需要一个请求体作为参数

xhr.send(body);  
xhr.send()就是直接把第二步open那个发送

**get**请求的请求参数直接在url地址设置即可

xhr.open("get","/testAJAX2?username=admin&password=123123");

**post**要写send中

**4. 接收响应格式：接收路由发过来的信息**

xhr.open("post","/testAJAX3")

xhr.send("username=sunwukong&password=abcd");

xhr.onreadystatechange = function () {

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

xhr.responseText

}};

**responseText**：接收文本格式的响应数据

var一个变量来接收下面的值，就可以用了  
 var text = xhr.responseXML接收xml格式的响应数  
 var text =xhr.responseText接收文本格式的响应数据

**readyState** 可以用来查看请求当前的状态  
0: 请求未初始化(xhr对象刚刚创建)  
1: 服务器连接已建立 (请求信息设置完毕，open()和send()调用完)  
2: 请求已接收 (服务器已经收到请求)  
3: 请求处理中 (服务器正在处理请求)  
4: 请求已完成，且响应已就绪  
**onreadystatechange**

这个事件会在 readyState 属性发生改变时触发

//3. 发送请求  
 xhr.send();  
 //4. //接收响应

onreadystatechange 这个事件会在 readyState 属性发生改变时触发

xhr.onreadystatechange = function () {  
 if(xhr.readyState ==4 && xhr.status ==200){  
 var text = xhr.responseText;  
 list.innerHTML += "<li>"+text+"</li>"  
 }  
 }  
 }  
}

**AJAX的例子**

在一个搭建好的web服务器中，在public添加一个html(0.1test.html)文件，

**1.首先书写body**

<button id="btn01">点我发送AJAX请求</button>  
<ul id="list"></ul>

**2.然后添加script代码**

window.onload = function () {  
 var btn01 = document.getElementById("btn01");  
 btn01.onclick = function () {  
 //1. 创建XMLHttpRequest对象  
 var xhr = new XMLHttpRequest();  
 //2. 设置请求的信息

xhr.open('get','/text?t='+Date.now());

**3.在路由中接收**

**4.运行**：

router.get('/text',function (req,res) {  
  
 console.log('收到请求');  
 res.send('text收到')  
  
})

在bin/www界面中dug运行一下（相当于我们之前的index，web框架的主界面）输入网址<http://localhost:3000/0.1test.html>

**get:ajax的发送响应写法**

**1.在body中：**

<button id="btn02">发送带参数的AJAX请求</button>

**2.script：**

var btn03 = document.getElementById("btn03");  
btn03.onclick = function () {  
 var xhr = new XMLHttpRequest();  
 xhr.open("post","/testAJAX3");

//这个比起get要多加一个头，并且数据要在send中发

xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");  
 xhr.send("username=sunwukong&password=abcd");   
 xhr.onreadystatechang = function () {  
 if(xhr.readyState == 4 && xhr.status == 200){  
 alert(xhr.responseText); } };};

**3. 路由：**  
post**:ajax的发送响应写法**

router.post("/testAJAX3" , function (req , res) {  
 console.log("testAJAX3 收到请求~~~~");  
 //接收请求参数  
 var username = req.body.username;  
 var password = req.body.password;  
 console.log(username , password);  
 res.send("testAJAX3 返回的响应");  
});

router.get("/testAJAX2", function (req , res) {  
 console.log("testAJAX2 收到请求~~~~")

//接收参数   
 var username = req.query.username;  
 var password = req.query.password;  
 console.log(username , password);  
 res.send("testAJAX2 返回的响应");  
});

var btn02 = document.getElementById("btn02");  
btn02.onclick = function () {  
 var xhr = new XMLHttpRequest();  
 xhr.open("get","/testAJAX2?username=admin&password=123123");  
 xhr.send();  
 xhr.onreadystatechang = function () {  
 if(xhr.readyState == 4 && xhr.status == 200){  
 alert(xhr.responseText);  
 } };};

**1.在body中：**

<button id="btn03">发送post请求</button>

**2.在script中：**  
**3.路由**  
**try—catch-finally**

**try{**

有可能出错的代码,没错正常出来，出错不会执行try的语句

执行catch这种的语句，不会影响后面的代码

**}catch**

catch(e){//e表示的是出错的原因和信息

会在try中的语句出错的时候执行

**}finally{**

finally中的代码总会执行，出不出错都执行

**}**

**注意：**必须有try，catch和finally必须有一个，也可以都有

**ajax的兼容性**

在老版本的IE中，不支持XMLHttpRequest，所以使用new XMLHttpRequest()会报错  
 IE6支持的方式  
 var xhr = new ActiveXObject("Msxml2.XMLHTTP") IE5.5以下支持的方式  
 var xhr = new ActiveXObject("Microsoft.XMLHTTP")

//自定义一个函数，用来获取xhr对象

**注意：加了这个函数之后，后面只要用到前者的都改成后者**

function getXhr() {  
 try {  
 return new XMLHttpRequest();  
 }catch (e){  
 try{  
 return new ActiveXObject("Msxml2.XMLHTTP");  
 }catch (e){  
 try{  
 return new ActiveXObject("Microsoft.XMLHTTP");  
 }catch (e){  
 alert("你是外星来的吧，你的浏览器不支持AJAX");  
 }  
 }  
 }}

var xhr = new XMLHttpRequest();  
var xhr = getXhr();

**ajax发送后，路由回复响应的方式**

**一、xml响应方式**

**1.以XML格式返回数据**

当我们解析xml格式的数据，需要通过xhr.responseXML来接收响应  
 该属性会返回一个对象，这个对象类似于DOM中的Document ,

**2.这个对象代表返回的整个xml文档**

以xml格式返回数据  
 <student><name>孙悟空</name><age>18</age>

</address></student>  
**3. 当响应头是一个xml格式的数据时，我们还必须设置一个响应头告诉浏览器响应的内容是一个xml：**

<student><name>admin</name><age></age></ student >

**3.当响应体是xml格式时的数据，我们还要必须设置一个响应头**

**XMl的使用例子：**

**1.在body中**：

<button id="btn04">加载XML格式的响应</button>

**2.在js中：**

var btn04 = document.getElementById("btn04");  
btn04.onclick = function () {   
 var xhr = new XMLHttpRequest();  
 xhr.open("get","/testAJAX4");

xhr.send();  
 xhr.onreadystatechange = function () {  
 if(xhr.readyState == 4 && xhr.status == 200){  
 //当我们解析xml格式的数据，需要通过xhr.responseXML来接收响应  
 //该属性会返回一个对象，这个对象类似于DOM中的Document , 这个对象代表返回的整个xml文档  
 var doc = xhr.responseXML;  
 //通过document 来对文档进行查询  
//<student><name>孙悟空</name><age>18</age><gender>男</gender><address>花果山</address></student>  
var nameEle = doc.getElementsByTagName("name")[0];  
var name = nameEle.firstChild.nodeValue;  
 var age = doc.getElementsByTagName("age")[0].firstChild.nodeValue;  
 var gender = doc.getElementsByTagName("gender")[0].firstChild.nodeValue;  
 var address = doc.getElementsByTagName("address")[0].firstChild.nodeValue;  
 alert(name +","+ age +","+ gender +","+ address);  
 }  
 };  
  
};

1. **路由：**

router.get("/testAJAX4",function (req , res) {  
 /\*1. 以xml格式返回数据

<student><name>孙悟空</name><age>18</age><gender>男</gender><address>花果山</address></student>  
 当响应头是一个xml格式的数据时，我们还必须设置一个响应头告诉浏览器响应的内容是一个xml  
 Content-type : application/xml  
 \*/  
 res.set("content-type","application/xml");  
 res.send("<student><name>孙悟空</name><age>18</age><gender>男</gender><address>花果山</address></student>");  
  
});

**二、json响应方式**

1.JSON格式的响应数据，需要使用xhr.responseText来接收

将json转换为js对象  
2.JSON.parse()用于将一个json字符串转换为一个js对象

JSON.stringify()用于将一个js对象转换为一个字符串  
 var obj = JSON.parse(json);

1. 当我们将一个对象设置响应体时，Express会自动将其转换为JSON发送

**1.在body中：**  
<button id="btn05">加载JSON格式的响应</button>

**2.在js中:**

var btn05 = document.getElementById("btn05");  
 btn05.onclick = function () {  
 var xhr = new XMLHttpRequest();  
 xhr.open("get","/testAJAX5");  
 xhr.send();  
 xhr.onreadystatechange = function () {  
 if(xhr.readyState == 4 && xhr.status == 200){  
 //JSON格式的响应数据，需要使用xhr.responseText来接收  
 var json = xhr.responseText;  
 //将json转换为js对象  
 //JSON.parse()用于将一个json字符串转换为一个js对象  
 //JSON.stringify()用于将一个js对象转换为一个字符串  
 var obj = JSON.parse(json);   
 alert(obj.age);   
 }  
 };  
 };  
};

**3.路由**：

router.get("/testAJAX5" , function (req , res) {  
 //创建一个对象  
 var obj = {  
 name:"孙悟空",  
 age:18,  
 gender:"男",  
 address:"花果山"  
 };  
 //当我们将一个对象设置响应体时，Express会自动将其转换为JSON发送  
 res.send(obj);  
});

jQuery的使用  
**$.ajax()**  
1. 他是jQuery的底层ajax方法，像$.post() $.get $.getJSON()都是基 于该方法一般除了有特殊的需求，不会使用$.ajax();  
2. 在ajax()可以直接传递一个配置对象，将ajax请求相关的配置在对象中设置

**type** 请求的类型 **url** 请求的地址

**例子：**

$.ajax({  
 type: "get",  
 url: "/testAJAX5",  
 data: "username=John&password=Boston",  
 dataType:"text",  
 success: function(data){  
 alert(data);

err:  
 }  
 });

**data** 要发送的请求参数（get请求的查询字符post请求的请求体）  
**success** 请求发送成功回调函数，返回的数据会以该回调函数的实参的形式返回  
**dataType** 设置期望返回的数据类型(text json ...)  
 如果不指定类型jQuery会自动判断

**post和get的方法：**  
 $.get(url, [data], [callback], [type])  
 $.post(url, [data], [callback], [type])  
 向服务器发送get或post请求  
 参数：  
**url** 请求地

**data** 请求参数(需要一个对象，会自动将对象转换为查询字符串或请求体)  
**callback** 请求发送完成后调用的函数  
**type** 期望返回值类型（相当于ajax中的dataType）

$.**post**("/testAJAX3",{username:"admin", password:"123123"} ,function (data) {  
 alert(data);  
},"text");

$("#btn01").click(function () {  
$.**get**("/testAJAX2",{

username:"admin",

password:"123123"

} ,function (data) {  
 alert(data);  
 },"text");  
  
});

**art-template的使用步骤** 1.下载并将template-web.js引入到当前页面中  
 2.创建模板（在网页内部来创建模板）  
 - art-template的模板需要创建在一个script标签中  
 3.渲染模板，并将渲染结果插入到页面中  
 template(id, data)

z

$("#btn01").click(function () {  
 //template()函数用来对模板进行渲染，它会根据模板的id来寻找模板  
 //并将数据插入进模板中对其进行渲染，最终将渲染后的html代码作为结果返回  
 var html = template("**temp01"** , {username:"猪八戒"});  
  
 //将渲染后的模板插入到网页中  
 $("body").append(html);  
  
});

在body中

<**script id="temp01" type="text/html"**></**script>**

**art-templated 语法**

**在body中**

**{{变量}}** 可以将一个内容在页面中输出 相当于 <%= %>  
**{{@变量}}** 可以将一个内容直接在页面中输出，不会转义，相当于 <%- %>

$("#btn01").click(function () {  
  
 $("body").append(template("temp01",{username:"<span>孙悟空</span>" , age:16 , list:["白骨精","蜘蛛精","玉兔精"]}));  
  
});

**条件判断**  
**{{if value}} ... {{/if}}  
{{if v1}} ... {{else if v2}} ... {{/if}}**

{{if age>=18}}  
 <h3>你已经成年了~</h3>  
{{else}}  
 <h3>你还未成年~</h3>  
{{/if}}

<h1>hello {{username}}</h1>  
<h1>hello {{@username}}</h1>

**遍历**  
  **{{each target}}  
 {{$index}} {{$value}}  
 {{/each}}**  
 - target 表示要遍历的数组  
 - $index 表示当前遍历到元素的索引  
 - $value 表示当前遍历到元素的值

<ul>  
 {{each list}}  
 <li>{{$index}}-{{$value}}</li>  
 {{/each}}  
</ul>