使用form 表单提交 是整个的页面 刷新

使用js ajax 表单提交:整个页面没有切换 仅仅是页面的局部的div刷新 异步刷新

**原生AJAX**

AJAX = Asynchronous JavaScript and XML（异步的 JavaScript 和 XML）。

**AJAX 不是新的编程语言，而是一种使用现有标准的新方法。即使用JS完成前后台通迅**。

AJAX 最大的**优点**是在不重新加载整个页面的情况下，可以与服务器交换数据并更新部分网页内容。

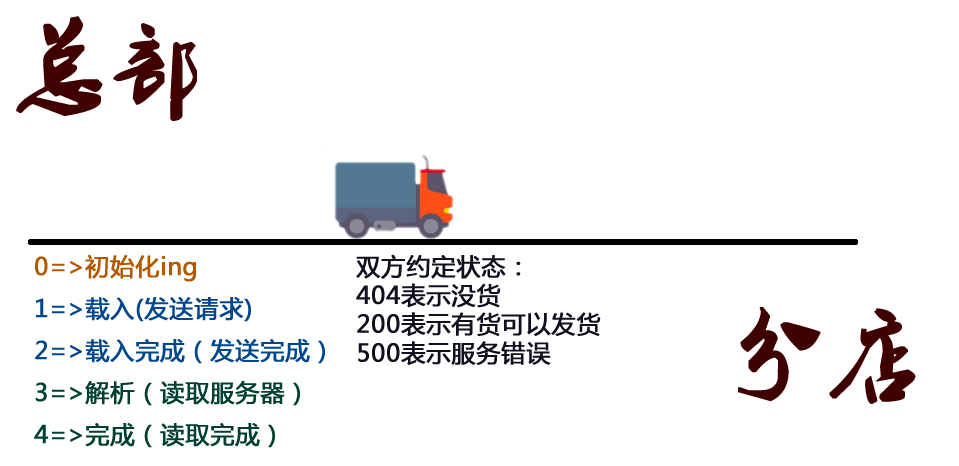
AJAX 不需要任何浏览器插件，但需要用户允许JavaScript在浏览器上执行。

**readyState（状态值）和status（状态码）**

readyState，是指运行AJAX所经历过的几种状态，无论访问是否成功都将响应的步骤，可以理解成为AJAX运行步骤，使用“ajax.readyState”获得 错误类似空车回来

status，是指无论AJAX访问是否成功，由HTTP协议根据所提交的信息，服务器所返回的HTTP头信息代码，使用“ajax.status”获得

**总体理解：**可以简单的理解为state代表一个整体的状态。而status是这个大的state下面具体的小的状态。



**什么是readyState**

readyState是XMLHttpRequest对象的一个属性，用来标识当前XMLHttpRequest对象处于什么状态。

readyState总共有5个状态值，分别为0~4，每个值代表了不同的含义

|  |  |
| --- | --- |
|  | 0：初始化，XMLHttpRequest对象还没有完成初始化  1：载入，XMLHttpRequest对象开始发送请求  2：载入完成，XMLHttpRequest对象的请求发送完成  3：解析，XMLHttpRequest对象开始读取服务器的响应  4：完成，XMLHttpRequest对象读取服务器响应结束 |

**什么是status**

status是XMLHttpRequest对象的一个属性，表示响应的HTTP状态码

在HTTP1.1协议下，HTTP状态码总共可分为5大类

|  |  |
| --- | --- |
|  | 1xx：信息响应类，表示接收到请求并且继续处理  2xx：处理成功响应类，表示动作被成功接收、理解和接受  3xx：重定向响应类，为了完成指定的动作，必须接受进一步处理  4xx：客户端错误，客户请求包含语法错误或者是不能正确执行  5xx：服务端错误，服务器不能正确执行一个正确的请求    100——客户必须继续发出请求  101——客户要求服务器根据请求转换HTTP协议版本  200——交易成功  201——提示知道新文件的URL  202——接受和处理、但处理未完成  203——返回信息不确定或不完整  204——请求收到，但返回信息为空  205——服务器完成了请求，用户代理必须复位当前已经浏览过的文件  206——服务器已经完成了部分用户的GET请求  300——请求的资源可在多处得到  301——删除请求数据  302——在其他地址发现了请求数据  303——建议客户访问其他URL或访问方式  304——客户端已经执行了GET，但文件未变化  305——请求的资源必须从服务器指定的地址得到  306——前一版本HTTP中使用的代码，现行版本中不再使用  307——申明请求的资源临时性删除  400——错误请求，如语法错误  401——请求授权失败  402——保留有效ChargeTo头响应  403——请求不允许  404——没有发现文件、查询或URl  405——用户在Request-Line字段定义的方法不允许  406——根据用户发送的Accept拖，请求资源不可访问  407——类似401，用户必须首先在代理服务器上得到授权  408——客户端没有在用户指定的饿时间内完成请求  409——对当前资源状态，请求不能完成  410——服务器上不再有此资源且无进一步的参考地址  411——服务器拒绝用户定义的Content-Length属性请求  412——一个或多个请求头字段在当前请求中错误  413——请求的资源大于服务器允许的大小  414——请求的资源URL长于服务器允许的长度  415——请求资源不支持请求项目格式  416——请求中包含Range请求头字段，在当前请求资源范围内没有range指示值，请求也不包含If-Range请求头字段  417——服务器不满足请求Expect头字段指定的期望值，如果是代理服务器，可能是下一级服务器不能满足请求  500——服务器产生内部错误  501——服务器不支持请求的函数  502——服务器暂时不可用，有时是为了防止发生系统过载  503——服务器过载或暂停维修  504——关口过载，服务器使用另一个关口或服务来响应用户，等待时间设定值较长  505——服务器不支持或拒绝支请求头中指定的HTTP版本 |

JSON和ajax

**什么是 JSON ？**

JSON 指的是 JavaScript 对象表示法（**J**ava**S**cript **O**bject **N**otation）

JSON 是轻量级的文本数据交换格式

JSON 独立于语言 \*

JSON 具有自我描述性，更易理解

**注：json文件不能写注释，必须用双引号，键和值都是双引号**

前后后通迅技术：

分离为：

1.前端获取到JSON,并且解析JSON

2.后台收到前端传递的数据，查询数据库，在php里有一段，json\_encode()编码，生成一段json，返回到前端。

再把第一步拆分成两块：

一块是ajax 拿到数据 json

另一块，解析数据，生成页面。

form 表单 action="abc.php" 整体页面刷新

ajajx 技术，获取后台传递的数据，可以局部的显示在某个div.

授课过程：

json数据一步步更复杂，数组，数组套数组，数组套对象。

直到php 输出 json

有时间讲下无刷新技术和页面整体刷新的区别

增删改查

dom操作

**前后台数据传输**

通过 **form表单**。提交数据。例如 get方式 （仅在地址栏后面加上数据）

通过**ajax传输**。页面无需刷新。前端承担了更多的工作量。大量减轻服务器的压力。

**前台向后台发送请求的方法：**

**GET和POST**是什么？**HTTP**协议中的两种发送请求的方法。

HTTP是什么？HTTP是基于TCP/IP的关于数据如何在万维网中如何通信的协议。

HTTP的底层是TCP/IP。所以GET和POST的底层也是TCP/IP，也就是说，GET/POST都是TCP链接。GET和POST能做的事情是一样一样的。你要给GET加上request body，给POST带上url参数，技术上是完全行的通的。

 怎么回事？



（1）**GET提交，请求的数据会附在URL之后**（就是把数据放置在HTTP协议头中），以?分割URL和传输数据，多个参数用&连接；例如：**login.php?name=hy**ddd&password=idontknow&verify=%E4%BD%A0%E5%A5%BD。如果数据是英文字母/数字，原样发送，如果是空格，转换为+，如果是中文/其他字符，则直接把字符串用BASE64加密，得出如：%E4%BD%A0%E5%A5%BD，其中％XX中的XX为该符号以16进制表示的ASCII。

　　**POST提交：把提交的数据放置在是HTTP包的包体中**。上文示例中红色字体标明的就是实际的传输数据

     因此，**GET提交的数据会在地址栏中显示出来，而POST提交，地址栏不会改变**

(2)**传输数据的大小：**首先声明：HTTP协议没有对传输的数据大小进行限制，HTTP协议规范也没有对URL长度进行限制。

而在实际开发中存在的限制主要有：

GET:特定浏览器和服务器对URL长度有限制，例如 IE对URL长度的限制是2083字节(2K+35)。对于其他浏览器，如Netscape、FireFox等，理论上没有长度限制，其限制取决于操作系 统的支持。

因此对于**GET提交时，传输数据就会受到URL长度的 限制(2K左右）**。

POST:由于不是通过URL传值，理论上数据不受限制。但实际各个WEB服务器会规定对post提交数据大小进行限制，Apache、IIS6都有各自的配置。

(3)**安全性：**

**POST的安全性要比GET的安全性高**。注意：这里所说的安全性和上面GET提到的“安全”不是同个概念。上面“安全”的含义仅仅是不作数据修 改，而这里安全的含义是真正的Security的含义，比如：通过GET提交数据，用户名和密码将明文出现在URL上，因为(1)登录页面有可能被浏览器缓存，(2)其他人查看浏览器的历史纪录，那么别人就可以拿到你的账号和密码了，除此之外，使用GET提交数据还可能会造成Cross-siterequest forgery攻击

（4）Http get,post,soap协议都是在http上运行的

1）get：请求参数是作为一个key/value对的序列（查询字符串）附加到URL上的

查询字符串的长度受到web浏览器和web服务器的限制（如IE最多支持2048个字符），不适合传输大型数据集同时，它很不安全

2）post：请求参数是在http标题的一个不同部分（名为entitybody）传输的，这一部分用来传输表单信息，因此必须将Content-type设置为:application/x-www-form-urlencoded。post设计用来支持web窗体上的用户字段，其参数也是作为key/value对传输。

但是：它不支持复杂数据类型，因为post没有定义传输数据结构的语义和规则。

3）soap：是http post的一个专用版本，遵循一种特殊的xml消息格式

Content-type设置为: text/xml   任何数据都可以xml化

T和POST还有一个重大区别，简单的说：

GET产生一个TCP数据包；POST产生两个TCP数据包。

于GET方式的请求，浏览器会把http header和data一并发送出去，服务器响应200（返回数据）；

而对于POST，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok（返回数据）。

也就是说，GET只需要汽车跑一趟就把货送到了，而POST得跑两趟，第一趟，先去和服务器打个招呼“嗨，我等下要送一批货来，你们打开门迎接我”，然后再回头把货送过去。

T和POST还有一个重大区别，简单的说：

GET产生一个TCP数据包；POST产生两个TCP数据包。

于GET方式的请求，浏览器会把http header和data一并发送出去，服务器响应200（返回数据）；

而对于POST，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok（返回数据）。

也就是说，GET只需要汽车跑一趟就把货送到了，而POST得跑两趟，第一趟，先去和服务器打个招呼“嗨，我等下要送一批货来，你们打开门迎接我”，然后再回头把货送过去。

**AJAX方法 （js原生）：**

使用js 封装好的对象，和对象的方法（函数），请求后台。

例如封装好的对象。data 是日期对象 math XMLHttpRequest是前台向后台请求的对象。

打开后台（路径），发送请求，监听请求，收到代号（状态码），判断成败。

function ajax(str){

// XMLHttpRequest对象 类似Date Math

var 请求对象 = new XMLHttpRequest()

console.log(请求对象.readyState); // 0

//get post

请求对象.open("get", str, true);

//发送请求

请求对象.send();

console.log(请求对象.readyState); // 1

/\*给请求对象，添加状态监听，状态改变会触发 \*/

请求对象.onreadystatechange = function () {

// console.log(请求对象.status); //HTTP状态吗

console.log(请求对象.readyState); //2 3 4

console.log(请求对象.status); //

if(请求对象.readyState === 4 && 请求对象.status === 200){

var obj=(请求对象.responseText);

console.log("这是后台输出＝＝＝"+obj);

one.innerHTML='<h1>'+obj+'</h1>';

}

};

}

ajax("a1.txt");

**JQ AJAX方法 ：**

**$.get("某后台路径"，参数，成功触发函数（后台所给数据）);**

$.get(url,params,callback);

**$.getJSON("某后台路径"，参数，成功触发函数（后台所给JSON）);**

$.getJSON(url,params,callback);

**$.post("某后台路径"，参数，成功触发函数（后台所给数据）);**

$.post(url,params,callback);

**$(元素).load("某页面.html");**

$.load(url,params,callback);

**其中的参数params可以是对象也可以是字串 id=8**

**$.ajax({url:"某后台路径",success:function(msg){...}})**

里面的参数比较多：

**url**:

要求为String类型的参数，（默认为当前页地址）发送请求的地址。

**type**:

要求为String类型的参数，请求方式（post或get）默认为get。注意其他http请求方法，例如put和delete也可以使用，但仅部分浏览器支持。

**data**:

要求为Object或String类型的参数，发送到服务器的数据。如果已经不是字符串，将自动转换为字符串格式。get请求中将附加在url后。防止这种自动转换，可以查看　　processData选项。对象必须为key/value格式，例如{foo1:"bar1",foo2:"bar2"}转换为&foo1=bar1&foo2=bar2。如果是数组，JQuery将自动为不同值对应同一个名称。例如{foo:["bar1","bar2"]}转换为&foo=bar1&foo=bar2。

**dataType**:

要求为String类型的参数，预期服务器返回的数据类型。如果不指定，JQuery将自动根据http包mime信息返回responseXML或responseText，并作为回调函数参数传递。可用的类型如下：

xml：返回XML文档，可用JQuery处理。

html：返回纯文本HTML信息；包含的script标签会在插入DOM时执行。

script：返回纯文本JavaScript代码。不会自动缓存结果。除非设置了cache参数。注意在远程请求时（不在同一个域下），所有post请求都将转为get请求。

json：返回JSON数据。

jsonp：JSONP格式。使用SONP形式调用函数时，例如myurl?callback=?，JQuery将自动替换后一个“?”为正确的函数名，以执行回调函数。

text：返回纯文本字符串。

要求为Function类型的参数，发送请求前可以修改XMLHttpRequest对象的函数，例如添加自定义HTTP头。在beforeSend中如果返回false可以取消本次ajax请求。XMLHttpRequest对象是惟一的参数。

            function(XMLHttpRequest){

               this;   //调用本次ajax请求时传递的options参数

            }

**success**：要求为Function类型的参数，请求成功后调用的回调函数，有两个参数。

         (1)由服务器返回，并根据dataType参数进行处理后的数据。

         (2)描述状态的字符串。

         function(data, textStatus){

            //data可能是xmlDoc、jsonObj、html、text等等

            this;  //调用本次ajax请求时传递的options参数

         }

**error**:

要求为Function类型的参数，请求失败时被调用的函数。该函数有3个参数，即XMLHttpRequest对象、错误信息、捕获的错误对象(可选)。ajax事件函数如下：

       function(XMLHttpRequest, textStatus, errorThrown){

          //通常情况下textStatus和errorThrown只有其中一个包含信息

          this;   //调用本次ajax请求时传递的options参数

       }

**contentType**：

要求为String类型的参数，当发送信息至服务器时，内容编码类型默认为"application/x-www-form-urlencoded"。该默认值适合大多数应用场合。

**contentType:"application/json",  //发送信息至服务器时内容编码类型。**

服务端已不能按传统的 post 表单提交接收数据了，发送参数时，只能向后台改善json格式的字串（**把json转成字串** JSON.stringify）

**async**:

要求为Boolean类型的参数，默认设置为true，所有请求均为异步请求。如果需要发送同步请求，请将此选项设置为false。注意，同步请求将锁住浏览器，用户其他操作必须等待请求完成才可以执行。

**cache**:

要求为Boolean类型的参数，默认为true（当dataType为script时，默认为false），设置为false将不会从浏览器缓存中加载请求信息。

**jsonp**：

要求为String类型的参数，在一个jsonp请求中重写回调函数的名字。该值用来替代在"callback=?"这种GET或POST请求中URL参数里的"callback"部分，例如{jsonp:'onJsonPLoad'}会导致将"onJsonPLoad=?"传给服务器。

等等其它属性和方法，因不用而省略。

**注：contentType:"application/json",  //发送信息至服务器时内容编码类型。**

服务端已不能按传统的 post 表单提交接收数据了，发送参数时，只能向后台改善json格式的字串（**把json转成字串** JSON.stringify）

$.ajax({

contentType: "application/json",

type:'POST',

data:JSON.stringify({

pageNo:1,

pageSize:12,

cateIds:[],

keyWord:'编织袋'

}),

url:'http://web-bg.wanhutong.com/a/wht/members/prod/prods/search',

success:function(msg){console.log(msg)},

error:function(msg){console.log(msg)}

})[移动端：.note](note://F32A8A36CCDC44B8AD75000734272E21)

**原生ajax方法：**

1.简单的get方法：

请求对象xmlHttpRequest 打开地址 发送请求 准备收信

var obj=new XMLHttpRequest();

obj.open("get","loginJs.jsp?user=qq");

obj.send();

obj.onreadystatechange=function(){ console.log(obj.responseText); }

2.post方法：增加设置（请求头） send参数

var json={user:'hello',pwd:'world'}

var obj=new XMLHttpRequest();

obj.open("post","loginJs.jsp");

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

obj.send("user="+JSON.stringify(json));

obj.onreadystatechange=function(){ console.log(obj.responseText); }

分页技术：

先全部拿到数据，再分页。

每次请求后台，拿当前页面数据。

两者的优缺点比较下。

使用：ES6 的模板字串语法，反引号，兼容不太好。需要把es6转化为es5.

es5 字串拼接，不支持换行，不太好用。

分页技术 ；

每页显示的条数 count 5

当前页 cur 1

总页数 all 100

例如 ：翻到第5页

数据应当是一个【】

第1页：0 -----》 4

第2页：5 -----》 9

第2页：10 -----》 14

(cur-1)\*count cur\*count

编组的截取 slice()

二级菜单和选项卡，都有鼠标经过和离开的事件

都是二级内容的显示和隐藏，当前的内容显示，其它内容隐藏。

区别:

二级菜单，

div结构：包含的关系 每个li 包含标题及二级菜单

标题单独一个ul li 内容单独一个ul li 并列关系

两个页面间的交互

1.通过后台。中转。

2.cookies session

3.h5 localstorage

**跨 域**

**同源策略**

在介绍跨域之前，我们需要先了解一下同源策略。那么什么是浏览器的同源策略呢？

同源

同源是指协议，域名，端口完全相同。

比如

http://www.example.com/index.do

http://www.example.com/login.do

而下面的任意一种都是不同源的。

协议不同

http://www.example.com/index.do

https://www.example.com/login.do

域名不同

https://www.example.com/index.do

https://subdomain.example.com/login.do

https://www.test.com/login.do

端口不同

http://www.example.com:80/index.do

http://www.example.com:8080/login.do

同源策略

同源策略（SOP，Same origin policy）是指作为浏览器最基本安全功能的一种约定，它对以下请求进行限制来降低受到XSS、CSFR等攻击的风险。

浏览器的存储

浏览器DOM对象和脚本

AJAX请求

那么现在可以来认识一下什么是跨域了。

跨域在平常写html时会发现比如

<script src="http://www.a.com/script/1.js"></script>

<img src="http://www.b.com/1.jpg">

<link href="http://www.c.com/1.css">

这种标签是不会遇到'跨域'问题的，严格上来说，这不是跨域，跨域是指在脚本代码中向非同源域发送HTTP请求，这只是跨站资源请求

那么我们可以试试看用这种跨站资源请求的方式来实现跨域HTTP请求

。解决ajax的跨域问题有两种方法：

　　一、jsop

　　二、XMLHttpRequest2中可以配合服务端来解决，在响应头中加入Access-Control-Allow-Origin:\*

**1、同源：**

　　同源策略是浏览器的一种安全策略，所谓同源是指，域名，协议，端口号完全相同

　　1.1目的：保护用户信息安全

　　1.2限制：cookie、localStorage和IndexDB无法读取

　　无法操作跨域的iframe里的dom元素

　　ajax请求不能发送

**2、跨域：**

不同源则为跨域

jsonp跨域，仅支持get 方式，使用post会比较麻烦。

**get()请求：**

$.ajax({

type: "get",

url: "你的请求地址",

dataType: 'jsonp', //【jsonp进行跨域请求 只支持get】

data:{ //【这里填写是传给服务端的数据 可传可不传 数据必须是json格式】

"a":"b",

"c":"d"

},

success: function(data) { //【成功回调】

console.log(data);

},

error: function(xhr, type) { //【失败回调】

}

});

**post()请求：**

【在进行跨域前，你需要知道的东西】：用ajax进行跨域请求，很多人就知道jsonp，然而很多人根本就不知道使用ajax跨域同时使用jsonp只能支持get方式请求，如果你进行post请求会执行error，并且返回readyState=4同时status=200

$.ajax({

type:"post",

url:"你的请求地址",

data:{ //【这里填写是传给服务端的数据 可传可不传 数据必须是json格式】

"a":"b",

"c":"d"

},

dataType:'json', //【这里要小心啊，不要用jsonp，一定是json】

crossDomain: true, //【这个很重要，一定要加】

success:function(result){

console.log(result);

},

error:function(result){

console.log(result);

}

});

（服务器端代码）：就是后台语言端(这里我以php为例)

在你执行所有操作之前加上它，使用的框架的建议放在框架的入口文件的最前面

header('Access-Control-Allow-Origin: \*');

header('Access-Control-Allow-Methods: POST');

header('Access-Control-Max-Age: 1000');