**ajax:**

前后端数据交互的一种技术。

Asynchronous JavaScript and XML

异步 js 和 xm

XML：存储数据的一种格式。'[]','{key:value}'

在ajax中的X，不止XML这一种数据格式

字符串->

数组

json

XML

**ajax的优点：**

传输获取数据 , 不用跳转页面，在本页面请求服务器，

做到实时验证。减少用户返工率并且优化用户体验。

缓解服务器压力

**ajax流程 ：**

1.创建ajax对象 var ajax = new XMLHttpRequest();

2.填写地址 get方式 ： ajax.open('get','php/get.php?user='+encodeURI(this.value)+'&'+new Date().getTime(),true);

Post方式： ajax.open('post','php/get.php’ ,true);

3.发送请求 get方式 ：ajax.send();

Post方式：请求头：ajax.setRequestHeader( 'Content-Type','application/x-www-form-urlencoded');

ajax.send('user='+this.value);

4.等待服务器响应 ajax.onload=function(){}

1. 通话 ajax.responseText就是请求后返回给前端的数据。

**编码转换：**

encodeURI('刘') 转换为url编码

decodeURI('%E5%88%98') 将url编码转回来

**POST:**

通过send直接发送到服务器

理论上来说体积是无限大（一般后端会做限制）

因为是直接发送到服务器，所以不会被浏览器缓存，也不会暴露用户的信息（相对GET安全）

必须在send之前设置一个请求头：ajax.setRequestHeader( 'Content-Type','application/x-www-form-urlencoded');

open的第二个参数不用拼接字段，字段拼接要放进send中

（ajax.send('user='+this.value);）

**eval()：**  eval( '('+ajax.responseText+')' ) 将不标准的json装换成对象格式

**New Function(,)**

alert((new Function('a,b','return a+b'))(1,2)); //3

(new Function('','return '+ajax.responseText))() 将不标准的json装换成对象格式，面试的时候回答这个

**Xml:**

responseXML:

将xml数据格式的字符串转成DOM元素（转成document节点）

DOC.getElementById('div1')不能获取到自定义标签

在XML中找自定义标签的内容：

chrome:

innerHTML/textContent

IE10以上:

textContent

IE9及以下浏览器:

Text

**状态码：**

Ajax.status，当ajax请求完成后会有两种情况，一种是成功，一种是失败

成功的范围：ajax.status >=200 && ajax.status <=299 || ajax.status == 304；

**onreadystatechange: （状态值改变事件）**

因为IE678不支持浏览器不支持onload用 onreadystatechange （所有浏览器都兼容）

监控请求步骤 ajax.readyState

**readyState 状态值 状态说明**(0)未初始化   
此阶段确认XMLHttpRequest对象是否创建，并为调用open()方法进行未初始化作好准备。值为0表示对象已经存在，否则浏览器会报错－－对象不存在。   
(1)载入   
此阶段对XMLHttpRequest对象进行初始化，即调用open()方法，根据参数(method,url,true)完成对象状态的设置。并调用send()方法开始向服务端发送请求。值为1表示正在向服务端发送请求。   
(2)载入完成   
此阶段接收服务器端的响应数据。但获得的还只是服务端响应的原始数据，并不能直接在客户端使用。值为2表示已经接收完全部响应数据。并为下一阶段对数据解析作好准备。   
(3)交互   
此阶段解析接收到的服务器端响应数据。即根据服务器端响应头部返回的MIME类型把数据转换成能通过responseBody、responseText或responseXML属性存取的格式，为在客户端调用作好准备。状态3表示正在解析数据。   
(4)完成   
此阶段确认全部数据都已经解析为客户端可用的格式，解析已经完成。值为4表示数据解析完毕，可以通过XMLHttpRequest对象的相应属性取得数据。   
概而括之，整个XMLHttpRequest对象的生命周期应该包含如下阶段：   
创建－初始化请求－发送请求－接收数据－解析数据－完成

只要监控ajax.readyState到4就说明整个流程完毕，就可以执行我们想要的代码。

如果将onreadystatechange放到send之前，那么在IE下是可以多监听一次（意义不大，因为只有显示为4的时候，才表明流程完毕）

**同步：**

当请求发送完成之后，send() 会挂起，一直等到服务器返回了相关信息（请求完全结束）之后，才会返回并继续执行后面的代码。

下一段代码，必须等待上一段代码执行完毕才会运行

**异步：**

当请求发送完成之后，send() 就会立即返回，继续执行send() 后面的代码。

下一段代码，不用等待上一段代码执行就可以运行了

**jquery的ajax**

$.ajax({

**url:**'php/get\_json2.php',

**data:**'user='+encodeURI($('#text').val()),

**type:**'get',

**dataType:**'json',

**success:**function(json){

console.log(eval('('+json+')'));

},

**error:**function(data){

//alert('你错了')

console.log(data);

}

});

使用get方式请求

$.get('php/get\_json.php','user='+encodeURI($('#text').val()),function(data){

console.log(data);

});

使用post方式请求

$.post('php/post.php','user='+$('#text').val(),function(data){

console.log(data);

});

获取的是json格式的数据（默认已经把json转成了对象）

$.getJSON('php/get\_json.php?user='+$('#text').val(),function(data){

console.log(data);

});

**上传文件**

new FormData：

将要上传的数据格式，转成二进制数据，这个时候就可以不用写请求头

FormData只有一个方法，append(key,value)

只要用了FormData，那么请求头都没必要写了

1.

var oF = new FormData();

2.

oF.append('file',oFile.files[0]);

3.

ajax.send(oF);

oFile.files[0]：

就是要上传文件的详细信息：

最后修改的时间

名字

大小

类型。。。

上传必须要用post请求方式来传输。

如果后端返回的内容有中文编码格式，那么直接输入到页面中就能变成中文了。

在file中，有一个对象：files（详细信息的列表）

files[0]就是我们要找的东西；

ajax的上传方式：

1.new FormData()

2.给这个对象append(key,value)

key：跟后端的要求走

value:file元素的files[0];

3.send(这个对象)

上传的进度：

ajax.upload.onprogress的这个事件来监听上传的进度（这个事件必须添加才能监听）

这个事件绑定函数中的ev，显示上传过程中的细节

比如：上传的总体积，每次上传的体积

ev.loaded 已上传大小

ev.total 文件总大小