关于AJAX == 异步js和XML：

AJAX 是一种用于创建快速动态网页的技术。

通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

AJAX是基于现有的Internet标准，并且联合使用它们：

* XMLHttpRequest 对象 (异步的与服务器交换数据)
* JavaScript/DOM (信息显示/交互)
* CSS (给数据定义样式)
* XML (作为转换数据的格式)

XMLHttpRequest 是 AJAX 的基础。

创建XMLHttpRequest对象：

首先考虑浏览器兼容问题：（IE5 IE6使用ActiveX对象）

var xmlhttp;

if (window.XMLHttpRequest) {

// IE7+, Firefox, Chrome, Opera, Safari 浏览器执行代码

xmlhttp=new XMLHttpRequest();

} else {

// IE6, IE5 浏览器执行代码

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

}

向服务器发送请求：

1. open（）

open(*method*,*url*,*async*)

* *method*：请求的类型；GET 或 POST
* *url*：文件在服务器上的位置
* *async*：true（异步）或 false（同步）

1. send（string）：仅用于post请求。

以下情况只能用post请求：

* 无法使用缓存文件（更新服务器上的文件或数据库）
* 向服务器发送大量数据（POST 没有数据量限制）
* 发送包含未知字符的用户输入时，POST 比 GET 更稳定也更可靠

简单的GET请求：

为了得到实时的结果，可以向URL里添加唯一的ID，如果想通过get方法发送信息，可以向URL里添加信息。

简单的POST请求：

如果需要像 HTML 表单那样 POST 数据，请使用 setRequestHeader() 来添加 HTTP 头。然后在 send() 方法中规定您希望发送的数据。

Open方法的url是服务器上文件的地址，当使用 async=true 时，请规定在响应处于 onreadystatechange 事件中的就绪状态时执行的函数。当使用 async=false 时，不要编写 onreadystatechange 函数，把代码放到 send() 语句后面即可。

AJAX服务器响应：

需要使用 XMLHttpRequest 对象的 responseText 或 responseXML 属性。

如果来自服务器的响应非XML形式:

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

如果响应是XML形式：  
xmlDoc=xmlhttp.responseXML;

txt="";

x=xmlDoc.getElementsByTagName("ARTIST");

for (i=0;i<x.length;i++) {

txt=txt + x[i].childNodes[0].nodeValue + "<br>";

}

document.getElementById("myDiv").innerHTML=txt;

Onreadystatechange事件：

每当 readyState 改变时，就会触发 onreadystatechange 事件。

readyState 属性存有 XMLHttpRequest 的状态信息。

|  |  |
| --- | --- |
| onreadystatechange | 存储函数（或函数名），每当 readyState 属性改变时，就会调用该函数。 |
| readyState | 存有 XMLHttpRequest 的状态。从 0 到 4 发生变化。   * 0: 请求未初始化 * 1: 服务器连接已建立 * 2: 请求已接收 * 3: 请求处理中 * 4: 请求已完成，且响应已就绪 |
| status | 200: "OK" 404: 未找到页面 |

xmlhttp.onreadystatechange=function() { if (xmlhttp.readyState==4 && xmlhttp.status==200) { document.getElementById("myDiv").innerHTML=xmlhttp.responseText; } }

使用回调函数：

如果您的网站上存在多个 AJAX 任务，那么您应该为创建 XMLHttpRequest 对象编写一个*标准*的函数，并为每个 AJAX 任务调用该函数。

function myFunction() { loadXMLDoc("/try/ajax/ajax\_info.txt",function() { if (xmlhttp.readyState==4 && xmlhttp.status==200) { document.getElementById("myDiv").innerHTML=xmlhttp.responseText; } }); }

showHint()函数：

当用户在输入框输入字符时，web与服务器进行通信。该函数又onkeyup触发。

Ajax数据库实例：

**实例解释 - showCustomer() 函数**

当用户在上面的下拉列表中选择某个客户时，会执行名为 "showCustomer()" 的函数。该函数由 "onchange" 事件触发：

ajax可用来与XML文件进行交互式通信

关于JSON：

对于 AJAX 应用程序来说，JSON 比 XML 更快更易使用：

**使用 XML：**

* 读取 XML 文档
* 使用 XML DOM 来循环遍历文档
* 读取值并存储在变量中

**使用 JSON：**

* 读取 JSON 字符串
* 用 eval() 处理 JSON 字符串

访问对象值：

可以用.号也可以用[]来访问并且进行修改。

循环对象：

用for in来循环

删除对象属性：delete

JSON.parse()

JSON 通常用于与服务端交换数据。

在接收服务器数据时一般是字符串。

我们可以使用 JSON.parse() 方法将数据转换为 JavaScript 对象。

Json不能存储date对象，可以启用JSON.parse 的第二个参数 reviver，一个转换结果的函数，对象的每个成员调用此函数。

不建议在json中使用函数。

JSON.stringify()

JSON 通常用于与服务端交换数据。

在向服务器发送数据时一般是字符串。

我们可以使用 JSON.stringify() 方法将 JavaScript 对象转换为字符串。

JSON 不能存储 Date 对象。

JSON.stringify() 会将所有日期转换为字符串。

JSON 不允许包含函数，JSON.stringify() 会删除 JavaScript 对象的函数，包括 key 和 value，可以在执行 JSON.stringify() 函数前将函数转换为字符串来避免以上问题的发生。

JSON的使用：

JSON 最常见的用法之一，是从 web 服务器上读取 JSON 数据（作为文件或作为 HttpRequest），将 JSON 数据转换为 JavaScript 对象，然后在网页中使用该数据。

由于 JSON 语法是 JavaScript 语法的子集，JavaScript 函数 eval() 可用于将 JSON 文本转换为 JavaScript 对象。

eval() 函数使用的是 JavaScript 编译器，可解析 JSON 文本，然后生成 JavaScript 对象。必须把文本包围在括号中，这样才能避免语法错误：

var obj = eval ("(" + txt + ")");