# AJAX

## 课程大纲

阶段一：AJAX简介

阶段二：JavaScript中AJAX的实现方式  
 阶段三：封装AJAX

阶段四：AJAX实战

## 阶段一：AJAX简介

### 什么是AJAX？

AJAX即“**A***synchronous***J***avascript****A****nd***X***ML*”（异步JavaScript和XML）

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 是与服务器交换数据并更新部分网页的艺术，在不重新加载整个页面的情况下。

传统的网页（不使用 AJAX）如果需要更新内容，必须重载整个网页页面。

本质是在HTTP协议的基础上以异步的方式与服务器进行通信

### 什么是异步？

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。

### AJAX常见应用场景

**应用场景如：百度检索**



**应用场景如：京东用户名验证**





**应用场景如：分页**



## 阶段二：JavaScript中AJAX的实现方式

**打电话：**

手机

拨号

说

听

**AJAX：**

创建对象：

标准模式：XMLHttpRequest

IE6： ActiveXObject("Microsoft.XMLHTTP");

//第一步：创建AJAX对象

var xhr;

if(window.XMLHttpRequest){

xhr = new XMLHttpRequest(); //标准模式

}else{

xhr = new ActiveXObject("Microsoft.XMLHTTP") //IE6

}

连接服务器 xhr.open(请求方式get或post,请求路径,是否异步);

//第二步：连接服务器

//Xhr.open(“GET”,”handler.php?random=”+Math.random()+”&name=张三&age=12”,true);

xhr.open("POST","handler.php",true);

//若是post请求则需要如下设置

var data = "name=张三&pwd=1234565";

//xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

发送请求 xhr.send(); //post请求需要传参数。Get请求不用传参数

xhr.send(data); //POST请求若有提交的数据则传

// Xhr.send(null);//get请求不需要传实参

接收响应的内容 xhr.onreadystatechange 事件（若异步则注册该事件）

xhr.onreadystatechange = function(){

if(xhr.readyState==4){ //读取完成（并不是成功）

if(xhr.status==200){ //状态码 代表成功

var data = eval(xhr.responseText)

alert(data[0].name);

}else{ //代表失败

alert("失败")

}

}

}

补充：

xhr.readyState 请求状态

readyState是一个只读属性，用一个整数和对应的常量，表示XMLHttpRequest请求当前所处的状态。

* 0，对应常量UNSENT，表示XMLHttpRequest实例已经生成，但是open()方法还没有被调用。
* 1，对应常量OPENED，表示send()方法还没有被调用，仍然可以使用setRequestHeader()，设定HTTP请求的头信息。
* 2，对应常量HEADERS\_RECEIVED，表示send()方法已经执行，并且头信息和状态码已经收到。
* 3，对应常量LOADING，表示正在接收服务器传来的body部分的数据，如果responseType属性是text或者空字符串，responseText就会包含已经收到的部分信息。
* 4，对应常量DONE，表示服务器数据已经完全接收，或者本次接收已经失败了。

xhr.responseText; 服务端返回的内容

Xhr.status HTTP状态码

status属性为只读属性，表示本次请求所得到的HTTP状态码，它是一个整数。一般来说，如果通信成功的话，这个状态码是200。

Xhr.status 返回的是数字

xhr.statusText 数字加描述

* 200, OK，访问正常
* 301, Moved Permanently，永久移动
* 302, Move temporarily，暂时移动
* 304, Not Modified，未修改
* 307, Temporary Redirect，暂时重定向
* 401, Unauthorized，未授权
* 403, Forbidden，禁止访问
* 404, Not Found，未发现指定网址
* 500, Internal Server Error，服务器发生错误

基本上，只有2xx和304的状态码，表示服务器返回是正常状态。

## 阶段三：封装AJAX

/\*

数据处理工具

\*/

function urlParmsHandler(obj){

var arr = []; //定义一个数组存放数据

/\*遍历对象的属性和值并添加的数组中\*/

for(var key in obj){

arr.push(key + "=" + obj[key]);

}

return arr.join("&");

}

/\*

ajax工具库：

method:请求方式，get或post;

url:请求的路径；

isAsync:是否异步请求，true或false

data:发送的请求数据

success:成功时候的要执行的函数。

error:请求失败的时候要执行的函数

\*/

function ajax(o){

//第一步：创建AJAX对象

var xhr;

if(window.XMLHttpRequest){

xhr = new XMLHttpRequest(); //标准模式

}else{

xhr = new ActiveXObject("Microsoft.XMLHTTP") //IE6

}

//和服务器建立连接

if(o.method.toLowerCase()=="get"){ //如果是get请求方式

if(o.data){ //如果有发送数据

o.url= o.url + "?random=" + Math.random() + "&" + urlParmsHandler(o.data);

xhr.open("get",o.url,o.isAsync)

}else{

o.url = o.url + "?random=" + Math.random();

xhr.open("get",o.url,o.isAsync)

}

///xhr.open(“get”,,true);

//发送请求

xhr.send();

}else{ //如果是post请求

xhr.open("post",o.url,o.isAsync);

xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded'); //模拟表单提交

//发送请求

xhr.send(urlParmsHandler(o.data));

}

//是否是异步

if(o.isAsync){

//注册状态监听事件

xhr.onreadystatechange = function(){

responsetHandler();

}

}else{

responsetHandler();

}

/\*处理响应\*/

function responsetHandler(){

if(xhr.readyState==4){ //读取完成（并不是成功）

if(xhr.status==200){ //状态码 代表成功

o.success(xhr.responseText);

}else{ //代表失败

o.error(xhr.statusText);

}

}

}

}

### 跨域：

AJAX无法实现跨域请求数据。

什么是跨域？

就是一个域名去请求另一个域名下的数据

比如：

部署在127.0.0.1:80下的项目 无法用AJAX去请求 部署在 127.0.0.1:8077 项目下的文件。

只要ip+端口不一样

原因：浏览器的 “同源策略”

什么是同源策略：

为了保证安全性，浏览器所遵守的一种约定。

就是部署在不同ip或端口下的两个项目无法使用AJAX实现数据传输。

怎样实现跨域请求？

JSONP

JSONP(JSON with Padding)是[JSON](http://baike.baidu.com/view/136475.htm" \t "http://baike.baidu.com/_blank)的一种“使用模式”，可用于解决主流浏览器的跨域数据访问的问题。由于同源策略，一般来说位于 server1.example.com 的网页无法与不是 server1.example.com的服务器沟通，而 HTML 的<script> 元素是一个例外。利用 <script> 元素的这个开放策略，网页可以得到从其他来源动态产生的 JSON 资料，而这种使用模式就是所谓的 JSONP。用 JSONP 抓到的资料并不是 JSON，而是任意的JavaScript，用 JavaScript 直译器执行而不是用 JSON 解析器解析

## 阶段四：AJAX实战

案例：百度检索

案例：登录验证

案例：瀑布流