**Ajax**

AJAX ：Asynchronous JavaScript and XML（异步的 JavaScript 和 XML）

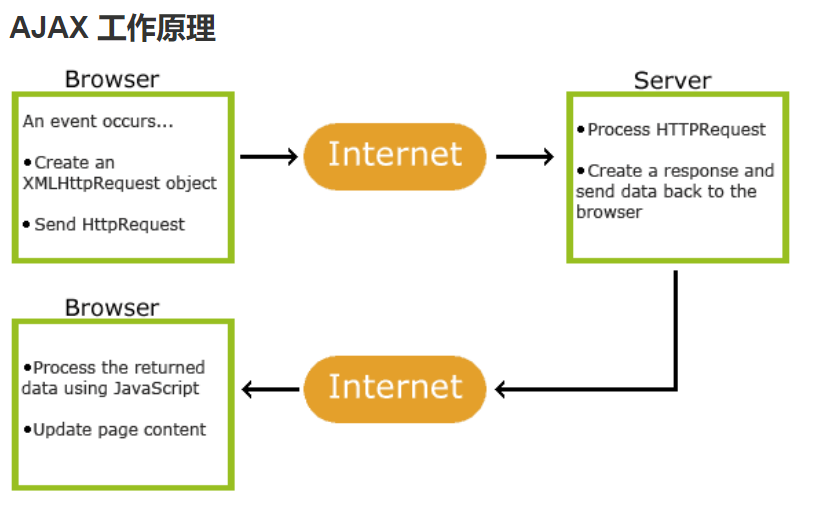
# Ajax简介

## 应用

AJAX 是一种在无需重新加载整个网页的情况下，能够更新部分网页的技术。

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。

## 原理



## 核心对象

XMLHttpRespond对象

# Ajax使用

## 完整代码

<script>

function loadXMLDoc()

{

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");

}

xmlhttp.onreadystatechange=function()

{

if (xmlhttp.readyState==4 && xmlhttp.status==200)

{

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

}

}

xmlhttp.open("GET","/try/ajax/ajax\_info.txt",true);

xmlhttp.send();

}

</script>

## XMLHttpRequest 对象

所有现代浏览器均支持 XMLHttpRequest 对象（IE5 和 IE6 使用 ActiveXObject）。

XMLHttpRequest 用于在后台与服务器交换数据。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。

### 创建对象（兼容性问题）

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");

}

### xmlHttpRequest对象属性



### xmlHttpRequest对象方法



### onreadystatechange 事件

当 readyState 等于 4 且status为 200 时，表示响应已就绪

xmlhttp.onreadystatechange=function() {

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

}

}

**注意：** onreadystatechange 事件被触发 4 次（0 - 4）, 分别是： 0-1、1-2、2-3、3-4，对应着 readyState 的每个变化。

## 服务器请求

//方法1：get

//url后面可以接参数

xmlhttp.open("GET","/try/ajax/demo\_get2.php?fname=Henry&lname=Ford",true);

xmlhttp.send();

//方法2：post

//调用方法setrequestheader来更改http头

xmlhttp.open("POST","/try/ajax/demo\_post2.php",true); xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded"); xmlhttp.send("fname=Henry&lname=Ford");

### get和post区别

与 POST 相比，GET 更简单也更快，并且在大部分情况下都能用。

然而，在以下情况中，请使用 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

### 异步

XMLHttpRequest 对象如果要用于 AJAX 的话，其 open() 方法的 async 参数必须设置为 true

通过 AJAX，JavaScript 无需等待服务器的响应，而是：

* 在等待服务器响应时执行其他脚本
* 当响应就绪后对响应进行处理

#### async为true

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

xmlhttp.onreadystatechange=function() {

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

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

}

xmlhttp.open("GET","/try/ajax/ajax\_info.txt",true);

xmlhttp.send();

#### async为false

我们不推荐使用 async=false，但是对于一些小型的请求，也是可以的。

请记住，JavaScript 会等到服务器响应就绪才继续执行。如果服务器繁忙或缓慢，应用程序会挂起或停止。

**注意：**当您使用 async=false 时，请不要编写 onreadystatechange 函数 - 把代码放到 send() 语句后面即可：

xmlhttp.open("GET","/try/ajax/ajax\_info.txt",false);

xmlhttp.send();

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

## 服务器响应

响应类型：



### respondText

如果来自服务器的响应并非 XML，请使用 responseText 属性。

responseText 属性返回字符串形式的响应，因此您可以这样使用：

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

### respondXML

如果来自服务器的响应是 XML，而且需要作为 XML 对象进行解析，请使用 responseXML 属性 。

**实例：**



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;

# ASP/PHP 实例

**ASP**即Active Server Pages，是Microsoft公司开发的服务器端脚本环境，可用来创建动态交互式网页并建立强大的web应用程序。当服务器收到对ASP文件的请求时，它会处理包含在用于构建发送给浏览器的HTML（Hyper Text Markup Language，超文本置标语言）网页文件中的服务器端脚本代码。

**PHP**即“超文本预处理器”，是一种通用开源脚本语言。PHP是在服务器端执行的脚本语言，与C语言类似，是常用的网站编程语言。PHP独特的语法混合了C、Java、Perl以及 PHP 自创的语法。利于学习，使用广泛，主要适用于Web开发领域。

## 例子

### html文件

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<script>

function showHint(str)

{

var xmlhttp;

if (str.length==0)

{

document.getElementById("txtHint").innerHTML="";

return;

}

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");

}

xmlhttp.onreadystatechange=function()

{

if (xmlhttp.readyState==4 && xmlhttp.status==200)

{

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

}

}

xmlhttp.open("GET","/try/ajax/gethint.php?q="+str,true);

xmlhttp.send();

}

</script>

</head>

<body>

<h3>在输入框中尝试输入字母 a:</h3>

<form action="">

输入姓名: <input type="text" id="txt1" onkeyup="showHint(this.value)" />

</form>

<p>提示信息: <span id="txtHint"></span></p>

</body>

</html>

### php文件

<?php

// Fill up array with names

$a[]="Anna";

$a[]="Brittany";

$a[]="Cinderella";

$a[]="Diana";

$a[]="Eva";

$a[]="Fiona";

$a[]="Gunda";

$a[]="Hege";

$a[]="Inga";

$a[]="Johanna";

$a[]="Kitty";

$a[]="Linda";

$a[]="Nina";

$a[]="Ophelia";

$a[]="Petunia";

$a[]="Amanda";

$a[]="Raquel";

$a[]="Cindy";

$a[]="Doris";

$a[]="Eve";

$a[]="Evita";

$a[]="Sunniva";

$a[]="Tove";

$a[]="Unni";

$a[]="Violet";

$a[]="Liza";

$a[]="Elizabeth";

$a[]="Ellen";

$a[]="Wenche";

$a[]="Vicky";

//get the q parameter from URL

$q=$\_GET["q"];

//lookup all hints from array if length of q>0

if (strlen($q) > 0)

{

$hint="";

for($i=0; $i<count($a); $i++)

{

if (strtolower($q)==strtolower(substr($a[$i],0,strlen($q))))

{

if ($hint=="")

{

$hint=$a[$i];

}

else

{

$hint=$hint." , ".$a[$i];

}

}

}

}

// Set output to "no suggestion" if no hint were found

// or to the correct values

if ($hint == "")

{

$response="no suggestion";

}

else

{

$response=$hint;

}

//output the response

echo $response;

?>

# XML实例

**异步加载XML文档**

function loadXMLDoc() {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

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

myFunction(this);

}

};

xhttp.open("GET", "cd\_catalog.xml", true);

xhttp.send();

}

function myFunction(xml) {

var i;

var xmlDoc = xml.responseXML;

var table="<tr><th>Artist</th><th>Title</th></tr>";

var x = xmlDoc.getElementsByTagName("CD");

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

table += "<tr><td>" +

x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +

"</td><td>" +

x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +

"</td></tr>";

}

document.getElementById("demo").innerHTML = table;

}

# 回调地狱

//以下代码,如果是异步传输数据,可能会报错

x = getData();

y = getMoreData(x);

z = getMoreData(y);

//解决办法

getData(function(x){

getMoreData(x,function(y){

getMoreData(y,function(z){

//回调层数太深,称为回调地狱

})

})

})

通过promise构造函数来解决回调地狱问题

//通过promise这个构造函数,创建一个对象

//promise对象的三种状态:pending,fulfill,reject

//promise构造函数,有一个参数,这个参数是一个回调函数,这个回调,接受两个参数,都能改变promise对象的状态

//两个参数:第一个参数,将pending==>fulfill;第二个参数,将pending==>reject

var promise = new Promise(function(resolve,reject){

if(/\* 成功 \*/){

resolve();

}else{

reject();

}

});

例子：

```js

// 判断生成的随机数是否是偶数

var promise = new Promise(function(resolve,reject){

setTimeout(function(){

var num = Math.floor(Math.random() \* 100);

if(num % 2 == 0){

resolve(num);

}else{

reject(num);

}

},3000);

});

promise.then(function(num){

console.log("resolve:" + num);

}).catch(function(num){

console.log("reject:" + num);

});

```