# Ajax02-Ajax异步编程

## 1 Ajax初窥

### 1.1 同步与异步

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。

同步：必须等待前面的任务完成，才能执行后面的任务；

异步：不受当前任务的影响。

现实生活中的一个例子：银行排队取款是同步，排队时玩手机是异步。

### 1.2 Ajax简介

当我们访问一个普通的网站时,当浏览器加载完:HTML,CSS,JS以后网站的内容就固定了.如果网站内容发生更改必须刷新页面才能够看到更新的内容。

异步更新的实际情况是：我们在访问新浪微博时,当你看到一大半了,会自动帮我们加载更多的微博,同时页面并没有刷新。

AJAX的作用就是：在浏览器中,我们也能够不刷新页面,通过ajax的方式去获取一些新的内容,类似网页有微博,朋友圈,邮箱等。

AJAX即 Asynchronous Javascript And XML（异步JavaScript和XML）

AJAX 不是一门的新的语言，而是对现有持术的综合利用，本质是在HTTP协议的基础上通过异步的方式与服务器进行通信。

### 1.3 Ajax示例

ajax使用的依旧是HTTP请求,那么HTTP请求需要什么：

请求的网址,方法get/post

提交请求内容数据,请求主体等

接收响应回来的内容

ajax的五步步骤：

1 建立XMLHTTPRequest对象

2 注册回调函数：当服务器回应我们了,我们想要执行什么逻辑

3 使用open方法设置和服务器端交互信息：提交的网址,数据

4 设置发送的数据，开始和服务器端交互，发送数据

5 更新界面：在注册的回调函数中,获取返回的数据,更新界面

var request = new XMLHttpRequest(); // 1-创建Ajax实例

request.open('get','./test.php',true); // 2-发起http请求

request.send(null);

request.onreadystatechange = function () { // 3-接受服务器端响应

console.log(request.readyState);

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

document.write(request.responseText);

}

}

## 2 Ajax常见API

### 2.1 Ajax简析

##### 2.1.1 请求组成

1. 请求行

request.open('get','./test.php',true);

第三个参数代表执行方式是否异步，默认为true（异步），这样不会阻塞代码。当我们参数写false，就是同步，那么这时候即使不写 if status ==\*\* 这样的判断也可以，即不需要监听状态，因为只有执行完请求后，才会执行后续操作，这时候已经拿到了后台返回的data或者err 可以直接console.log（res.responseText）

1. 请求头

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

3-请求主体

request.send(null);

仅用于POST请求。

##### 2.1.2 响应组成

1-获取状态行

request.status; // 获取状态码 200 OK 404 未找到 5开头 服务器错误

request.statusText; // 获取状态信息

2-获取响应头

request.getAllResponseHeaders(); //获取全部响应头

request.getResponseHeader(); //获取MIME类型响应头

1. 响应主体

request.responseText; //获得字符串形式的响应数据

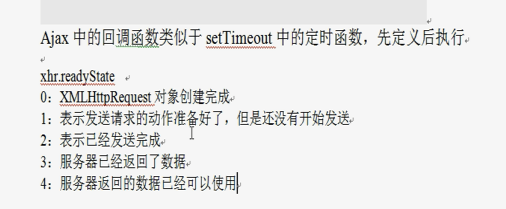
request.responseXML; //获得XML形式的响应数据

我们需要检测并判断响应头的MIME类型后确定使用request.responseText或者request.responseXML。

注意：文件的类型不是由文件的后缀名决定的。

1. readyState

每当readyState属性改变，就会调用onreadystatechange函数。



### 2.2 GET和POST

##### 2.2.1 GET和POST区别

GET没有请求主体，使用request.send(null)

GET可以在请求URL上添加请求参数，POST可以通过request.send('name=zs&age=10')

POST需要设置send（send发送参数数据，必须设置，没有则可以不写，如果有参数数据，没有写头信息，则服务器无法正确解码参数信息）

get请求时，send参数可以写null，也可以不写



##### 2.2.2 GET缓存问题

get请求会引起缓存问题，这里加入时间戳解决。

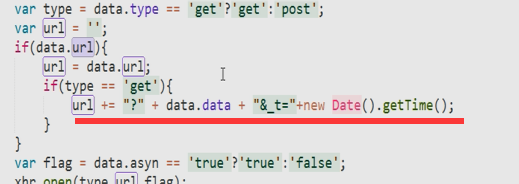
    缓存问题的解决：

        如果每次访问同一个url，那么后面的访问都会从缓存中获取，如果url中包含请求，刷新后的页面不会因为后台数据的改变而改变。

        解决办法：在url地址中添加一个&数据，这个数据在每次url访问的时候都会产生变化，即可。如使用随机数、时间戳等：

        '\*\*\*\*.php?username=lisi&password=123&' + new Date().getTime()

        POST在提交数据的时候没有缓存问题的存在。



##### 2.2.3 GET请求

var xml = new XMLHttpRequest(); // 创建XMLHttpRequest 对象

xml.open('get','01.ajax.php?name=fox'); // 设置跟服务端交互的信息

xml.send(null); // get请求这里写null即可

xhr.onreadystatechange = function () { // 接收服务器反馈

// 这步为判断服务器是否正确响应

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

alert(xml.responseText);

}

};

##### 2.2.4 POST请求

如果想要像form表单提交数据那样使用POST请求,需要使用XMLHttpRequest对象的setRequestHeader()方法 来添加 HTTP 头。然后在 send() 方法中添加想要发送的数据：

var xhr = new XMLHttpRequest(); // 异步对象

xhr.open('post', '02.post.php' ); // 设置属性

// 如果想要使用post提交数据,必须添加

xhr.setRequestHeader("Content-type","application/x-www-form-urlencoded");

xhr.send('name=fox&age=18'); // 将数据通过send方法传

xhr.onreadystatechange = function () { // 发送并接受返回

// 这步为判断服务器是否正确响应

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

alert(xhr.responseText);

}

};

### 2.4 文件上传的表单书写

|  |
| --- |
| <**form action="file\_big.php" method="post" enctype="multipart/form-data"**>  <**input type="file" name="upFile"** >  <**input type="submit"** > </**form**> |

### 2.5 XMLHttpRequest兼容

新版浏览器创建方式：

var xml=new XMLHttpRequest();

IE5/IE6创建方式：

var xml=new ActiveXObject("Microsoft.XMLHTTP");

兼容写法：

var request ;

if(XMLHttpRequest){

request = new XMLHttpRequest();

}else{

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

}

## 3 Ajax传输XML

### 3.1 XML简介

XML是一种标记语言，很类似HTML，其宗旨是用来传输数据，具有自我描述性，其格式要求如下：

必须有一个根元素

不可有空格、不可以数字或.开头、大小写敏感

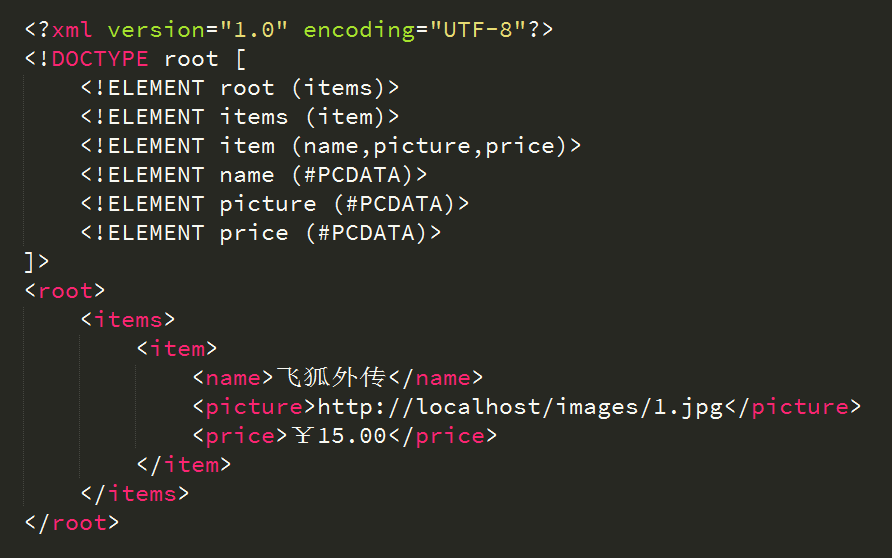
不可交叉嵌套

属性双引号

特殊符号要使用实体

注释和HTML一样

示例



### 3.2 XML语法

XML声明

<?xml version="1.0" encoding="UTF-8"?>

第一行是XML的声明,指定XML版本(1.0)以及使用的编码

根节点

XML中必须有一个根节点,所有的子节点都放置在根节点下

<root>

<name></name>

</root>

自定义标签

XML中没有默认的标签,所有的标签都是我们定义者自定义的

如：<fox></fox> <name></name>

XML中没有单标签,都是双标签

XML属性

跟HTML一样,XML的标签里面也能够添加属性type = 'text',但是不建议这样用,而是使用标签的方式来表述内容(下半部分代码)

<!-- 使用属性配合标签表述信息 -->

<person sex="female">

<firstname>Anna</firstname>

<lastname>Smith</lastname>

</person>

<!-- 使用标签来表述信息 -->

<person>

<sex>female</sex>

<firstname>Anna</firstname>

<lastname>Smith</lastname>

</person>

### 3.3 XML对象获取

因为XML就是标签,所以直接用解析Dom元素的方法解析即可

HTML代码

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

</head>

<body>

<person id='personXML'>

<name>fox</name>

<age>18</age>

<skill>小花花</skill>

</person>

</body>

</html>

获取代码

<script type="text/javascript">

var xmlObj = document.getElementById("personXML");

var name = xmlObj.getElementsByTagName('name')[0].innerHTML;

console.log(name);

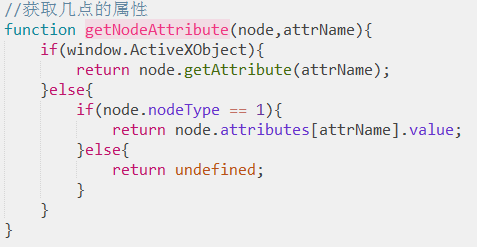
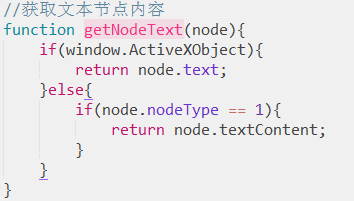
</script>

### 3.4 PHP设置header

在php中如果要使用xml传输数据,需要使用header()设置返回的内容为xml

header('content-type:text/xml;charset=utf-8');

### 3.5 XML解析



## 4 Ajax传输JSON

### 4.1 JSON简介

JSON即 JavaScript Object Notation，另一种轻量级的文本数据交换格式

语法规则:

数据在键值对中

数据由逗号分隔

花括号保存对象

方括号保存数组

数据类型: 无论 键 值 都是用双引号包起来

数字（整数或浮点数）

字符串（在双引号中）

逻辑值（true 或 false）

数组（在方括号中）

对象（在花括号中）

null

// 基本对象

{

"name":"fox",

"age":"18",

"sex":"true",

"car":null

}

// 数组

[

{

"name":"小小胡",

"age":"1"

},

{

"name":"小二胡",

"age":"2"

}

]

### 4.2 JS解析JSON

json字符串：var jsonStr = ‘{“name”:”三国”,”category”:”文学”}’;

json对象： var jsonObj = JSON.parse(jsonStr);

json对象转字符串： var s = JSON.stringify(jsonObj);

JSON解析方法：eavl()、JSON对象 JSON.parse()、JSON.stringify()

1-使用JSON对象解析：

JSON.parse()方法:将JSON字符串转化为JavaScript对象

JSON.stringify()方法:将JavaScript对象,转化为JSON字符串

2-IE8以下解析：

IE8以下没有JSON对象，使用第三方框架JSON2.js

var Obj = {

name:"fox",

age:18,

skill:"撩妹"

};

var jsonStr = JSON.stringify(Obj); // 将JavaScript对象格式化为JSON字符串

var jsonObj = JSON.parse(jsonStr); // 将JSON字符串转化为JavaScript对象

1. eval()方法解析

需要将内容使用()括号包裹。

var jsonObj = eval('('+jsonStr+')');

注意：使用eval()也可以将json字符串转换为json对象：var jsonObj = eval(“(“+ str +”)”);不推荐使用eval，因为如果传入的字符串是类似alert(123)这样的字符串，就会执行相应的函数。

### 4.3 PHP解析JSON

\* \*\*json\_decode()\*\*方法: 将`json`字符串转化为变量

\* \*\*json\_encode()\*\*方法: 将变量转化为`json`字符串

<?php

header("Content-Type:text/html;charset=utf-8");

// json字符串

$jsonStr = '{"name":"itcast","age":54,"skill":"歌神"}';

// 字符串转化为 php对象

print\_r(json\_decode($jsonStr));

echo "<br>";

// php数组

$arrayName = array('name' =>'littleFox' ,'age' => 13 );

// php对象 转化为 json字符串

print\_r(json\_encode($arrayName));

?>

输出结果为：

stdClass Object ( [name] => itcast [age] => 54 [skill] => 歌神 )

{"name":"littleFox","age":13}

## 5 Ajax封装

### 5.1 封装一：抽取公共部分

function ajax(method,url,data,success) {

// 创建对象

var xmlRequest ;

if (XMLHttpRequest) {

xmlRequest = new XMLHttpRequest();

}else{

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

}

// open方法

xmlRequest.open(method,url)

// 如果是post设置HTTP协议头

if (method=="post") {

xmlRequest.setRequestHeader("Content-type","application/x-www-form-urlencoded");

}

// send方法

xmlRequest.send(data);

// 判断状态改变,调用方法

xmlRequest.onreadystatechange = function () {

// 这步为判断服务器是否正确响应

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

success(ajax获取的数据);

}

};

}

### 5.2 封装二：优化参数

function ajax(option) {

// 创建对象

var xmlRequest ;

if (XMLHttpRequest) {

xmlRequest = new XMLHttpRequest();

}else{

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

}

// open方法

xmlRequest.open(option.method,option.url)

// 如果是post设置HTTP协议头

if (option.method=="post") {

xmlRequest.setRequestHeader("Content-type","application/x-www-form-urlencoded");

}

// send方法

xmlRequest.send(option.data);

// 判断状态改变,调用方法

xmlRequest.onreadystatechange = function () {

// 这步为判断服务器是否正确响应

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

option.success(ajax获取的数据);

}

};

}

### 5.3 封装三：参数格式化

function ajax(option) {

var xmlRequest ;

if (XMLHttpRequest) {

xmlRequest = new XMLHttpRequest();

}else{

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

}

// 格式化传入的数据为name=fox&age=18这样的格式

var formatStr = "";

for(var item in option.data){

// 获取属性名,属性值,进行拼接

formatStr+=item;// 属性名

formatStr+="=";//等号

formatStr+=option.data[item];//属性值

formatStr+="&";//分隔符

}

// 去除最后一个&

formatStr = formatStr.slice(0,-1);

// open方法 如果是get方法,那么url之后需要添加数据

if(option.method == "get"){

option.url = option.url+"?"+formatStr;

option.data = null;

}

// 调用open方法

xmlRequest.open(option.method,option.url)

// 如果是post设置HTTP协议头

if (option.method=="post") {

xmlRequest.setRequestHeader("Content-type","application/x-www-form-urlencoded");

}

// send方法 这里的data已经是修改过的,如果使用的是get方法,那么data为null

xmlRequest.send(option.data);

// 判断状态改变,调用方法

xmlRequest.onreadystatechange = function () {

// 这步为判断服务器是否正确响应

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

option.success(这里的数据是ajax获取的);

}

};

}

### 5.4 封装四：命名空间

为了避免自定义的Ajax方法被用户自定义的方法覆盖,添加到命名空间下,进行最后一步优化。

命名空间:为了防止自定义的Ajax函数被用户覆盖,可以通过将该函数设置为一个对象的方法。使用时,只需要通过对象.方法即可使用。

//定义命名空间

var fox = {

sayHi:function(name){

console.log(name);

}

}

//调用

fox.sayHi()

## 6 jQuery中的Ajax

### 6.1 示例

$.ajax({

url: './test.php',

type: 'get',

contentType: 'application/x-www-form-urlencoded',

data: null,

success: function (data) {

document.write(data);

}

});

### 6.2 四种请求方式

##### 6.2.1 $.ajax();

返回创建的XMLHttpRequest对象，ajax()只有一个参数，ke/value对象，包含各个配置及回调函数信息，如果指定了dataType，需要确保服务器返回正确的MIME信息。

$.ajax({

type: "post",

dataType: "html",

url: '/Resources/GetList.ashx',

data: dataurl,

success: function (data) {

if (data != "") {

$("#pager").pager({ pagenumber: pagenumber, pagecount: data.split("$$")[1], buttonClickCallback: PageClick });

$("#anhtml").html(data.split("$$")[0]);

}

}

});

##### 6.2.2 GET方式

这是一个简单的GET请求功能，以取代复杂的$.ajax，如果需要在出错时使用回调函数，仍然需要使用$.ajax。

$.get(

"test.cgi",

{ name: "John", time: "2pm" },

function(data){

alert("Data Loaded: " + data);

}

);

参数列表

url 必需。规定将请求发送的哪个 URL。

data 可选。规定连同请求发送到服务器的数据。

success(response,status,xhr) 可选。规定当请求成功时运行的函数。

额外的参数：

response 包含来自请求的结果数据

status 包含请求的状态

xhr 包含 XMLHttpRequest 对象

dataType 可选。规定预计的服务器响应的数据类型。

默认地，jQuery 将智能判断。

可能的类型："xml"、"html"、"text"、"script"、"json"、"jsonp"

##### 6.2.3 POST方式

这是一个简单的POST请求功能，以取代复杂的$.ajax，如果需要在出错时使用回调函数，仍然需要使用$.ajax。

$.post("/Resources/addfriend.ashx", { "fid": fids, "fname": fnames, "tuid": tuids, "tuname": tunames }, function (data) {

if (data == "ok") {

alert("添加成功！");

}

})

##### 6.2.4 GET请求载入JSON

$.getJSON("http://api.flickr.com/services/feeds/photos\_public.gne?tags=cat&tagmode=any&format=json&jsoncallback=?",

function(data){

$.each(data.items, function(i,item){

$("<img/>").attr("src", item.media.m).appendTo("#images");

if ( i == 3 ) return false;

});

});

### 6.3 格式化表单

我们在向服务器提交数据时,如果使用的是Ajax需要手动将数据格式化name=fox&age=18类似这样的格式,JQuery已经帮助我封装好了一个格式化数据的方法。

$(selector).serialize() 直接可以将form中拥有name属性的表单元素的字,进行格式化。

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>测试jq\_serialize方法</title>

<script type="text/javascript" src="./jquery.js"></script>

<script type="text/javascript">

$(function(){

$("#getFormInfo").on("click",function(){

var info = $("#testForm").serialize()

console.log(info);

})

})

</script>

</head>

<body>

<form id="testForm">

<input type="text" placeholder="您的姓名" name="userName">

<input type="text" placeholder="您的爱好" name="userHabbit">

<input type="text" placeholder="您最喜爱的食物" name="userHabbit">

</form>

<input type="button" value="格式化表单数据" id="getFormInfo">

</body>

</html>