# 尚马教育 JAVA WEB课程

# Ajax

文档编号：B16

创建日期： 2017-06-13

最后修改日期：2019-09-23

版 本 号：V2.0

电子版文件名：尚马教育-第二阶段-6.ajax.docx

**文档修改记录：**

|  |  |  |  |
| --- | --- | --- | --- |
| 更新日期 | 更新作者 | 更新说明 | 版本号 |
| 2018-08-01 | 王绍成 | 初始版本 | V1.0 |
| 2019-09-13 | 郭雪岩 | 内容更新 | V2.0 |

**主讲人：**

**郭雪岩**

目录

[1. Ajax](#_Toc30042_WPSOffice_Level1) [3](#_Toc30042_WPSOffice_Level1)

[1.1. 什么是ajax](#_Toc11163_WPSOffice_Level2) [3](#_Toc11163_WPSOffice_Level2)

[1.2. Ajax原理](#_Toc19508_WPSOffice_Level2) [3](#_Toc19508_WPSOffice_Level2)

[1.3. 原生ajax代码步骤](#_Toc11103_WPSOffice_Level2) [4](#_Toc11103_WPSOffice_Level2)

[1.4. JqueryAjax](#_Toc15658_WPSOffice_Level2) [6](#_Toc15658_WPSOffice_Level2)

## Ajax

### 什么是ajax

* AJAX指异步的JavaScript及XML（Asynchronous JavaScript And XML）
* AJAX 不是编程语言，而是一系列技术的结合，为请求数据方式提供了一种不同的模式

普通数据请求方式：

a标签跳转、用户自己输入浏览器地址栏、js中location.href、form提交，

模式：服务器接收到请求后，把页面需要展示的数据传回浏览器，浏览器使用这些新的数据完整的替换掉旧的页面数据。

Ajax数据请求方式：

Ajax使用XMLHttpRequest 对象发送请求

模式：服务器接收到请求后，把页面需要展示的数据传回浏览器，浏览器读取到这些数据，但是不使用这些数据直接替换掉页面之前的数据，而是允许设置一段js来读取返回的数据，由这段可以自由编辑的js内容决定返回的数据如何处理（一般根据返回数据制作一些页面特效或者修改旧页面中的部分数据），旧的页面数据没有被完整替换。

总结：ajax是一种新的与后台交互模式，一定程度上需要后台部分的配合

### Ajax原理

浏览器中的线程状态：

js是单线程的，那么是否代表参与js执行过程的线程就只有一个？

不是的，会有四个线程参与该过程，但是永远只有JS执行引擎线程在执行JS脚本程序，其他的三个线程只协助，不参与代码解析与执行。参与js执行过程的线程分别是：

**JS执行引擎线程**： 也称为JS内核，负责解析执行Javascript脚本程序的主线程（例如V8引擎）

**事件触发线程**： 归属于浏览器内核进程，不受JS引擎线程控制。主要用于控制事件（例如鼠标，键盘等事件），当该事件被触发时候，事件触发线程就会把该事件的处理函数推进事件队列，等待JS引擎线程执行

**定时器触发线程**：主要控制计时器setInterval和延时器setTimeout，用于定时器的计时，计时完毕，满足定时器的触发条件，则将定时器的处理函数推进事件队列中，等待JS引擎线程执行。  
注：W3C在HTML标准中规定setTimeout低于4ms的时间间隔算为4ms。

**HTTP异步请求线程**：通过XMLHttpRequest连接后，通过浏览器新开的一个线程，监控readyState状态变更时，如果设置了该状态的回调函数，则将该状态的处理函数推进事件队列中，等待JS引擎线程执行。  
注：浏览器对通一域名请求的并发连接数是有限制的，Chrome和Firefox限制数为6个，ie8则为10个。

总结：永远只有JS引擎线程在执行JS脚本程序，其他三个线程只负责将满足触发条件的处理函数推进事件队列，等待JS引擎线程执行。

ajax由js编写，使用XMLHttpRequest 对象对数据收发进行封装，

发送ajax请求时，ajax会要求调用方（浏览器）开启新线程向服务器进行请求，并对响应的结果进行反复读取，当读取到特定状态后，停止反复读取。读取过程中会把需要运行的js（回调函数）添加入js的执行队列（js执行引擎为单线程），js执行引擎按队列顺序执行。

### 原生ajax代码步骤

* 创建XMLHttpRequest对象
* 设置回调函数
* 初始化XMLHttpRequest组件（设置）
* 发送请求

|  |
| --- |
| //原生ajax调用过程  function myAjaxReq(){  var xmlhttp;  if (window.XMLHttpRequest){// code for IE7+, Firefox, Chrome, Opera, Safari  xmlhttp=new XMLHttpRequest();  }  else{// code for IE6, IE5  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");  }    //回调函数 当请求正常返回时 用js操作页面  xmlhttp.onreadystatechange=function(){  //readyState ajax请求的状态 4代表请求发送完成  //status http的状态 200代表正确返回  if (xmlhttp.readyState==4 && xmlhttp.status==200){  console.log(xmlhttp.responseText);  document.getElementById("showspan").innerHTML=xmlhttp.responseText;  }  }  //设置请求的地址 方式 是否使用异步  xmlhttp.open("POST","testAjax",true);  //设置请求头  xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");  //设置发送的参数  xmlhttp.send("uname="+document.getElementById("username").value);  } |

* XMLHttpRequest对象的属性和方法

|  |
| --- |
|  |
|  |

注意：使用ajax时参数需要自己传递，get方式把参数直接拼接在url后，post方式把参数写在send方法的参数中。

### JqueryAjax

原生ajax配置格式不规整，使用使用时经常使用jquery优化后的ajax

|  |
| --- |
| //jquery封装的ajax请求  //通过json格式设置参数  $.ajax({  url:"ajaxServlet",//地址  type:"post",//请求类型  data:{"uname":$("#username").val()}, //请求的参数 常用json格式  dataType:"text",//返回数据的类型  success:function(data){//响应成功时的回调函数 data表示返回的数据  console.log(data);  }  }); |

Jquery中简化ajax，json格式改为固定参数

|  |
| --- |
|  |

进一步简化

|  |
| --- |
| .getJSON(  “发送地址”,  发送参数,  回调函数  ) |

注意1：使用ajax时，需要自己拼接参数，可以使用$("#myform").serialize(),把form序列化成键值对字符串格式。

注意2：后台可以借助fastJSON工具，给前台返回json格式字符串。