# AJAX 教程

AJAX = Asynchronous JavaScript and XML（异步的 JavaScript 和 XML）。

AJAX 不是新的编程语言，而是一种使用现有标准的新方法。

AJAX 最大的优点是在不重新加载整个页面的情况下，可以与服务器交换数据并更新部分网页内容。

AJAX 不需要任何浏览器插件，但需要用户允许JavaScript在浏览器上执行。

## AJAX 应用

运用XHTML+CSS来表达资讯；

运用JavaScript操作DOM（Document Object Model）来执行动态效果；

运用XML和XSLT操作资料;

运用XMLHttpRequest或新的Fetch API与网页服务器进行异步资料交换；

注意：AJAX与Flash、Silverlight和Java Applet等RIA技术是有区分的。

# AJAX 简介

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。

## 您应当具备的基础知识

在继续学习之前，您需要对下面的知识有基本的了解：

* HTML / XHTML
* CSS
* JavaScript / DOM

如果您希望首先学习这些项目，请在我们的[首页](http://w3cschool.cc/" \o "W3CSchool 在线教程)访问这些教程。

## 什么是 AJAX ？

AJAX = 异步 JavaScript 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。

传统的网页（不使用 AJAX）如果需要更新内容，必需重载整个网页面。

有很多使用 AJAX 的应用程序案例：新浪微博、Google 地图、开心网等等。

## AJAX 工作原理



## AJAX是基于现有的Internet标准

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)

IMG_257  AJAX应用程序与浏览器和平台无关的！

## Google Suggest

在 2005 年，Google 通过其 Google Suggest 使 AJAX 变得流行起来。

Google Suggest 使用 AJAX 创造出动态性极强的 web 界面：当您在谷歌的搜索框输入关键字时，JavaScript 会把这些字符发送到服务器，然后服务器会返回一个搜索建议的列表。

# AJAX - 创建 XMLHttpRequest 对象

XMLHttpRequest 是 AJAX 的基础。

## 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 XMLHttpRequest 对象

所有现代浏览器（IE7+、Firefox、Chrome、Safari 以及 Opera）均内建 XMLHttpRequest 对象。

创建 XMLHttpRequest 对象的语法：

*variable*=new XMLHttpRequest();

老版本的 Internet Explorer （IE5 和 IE6）使用 ActiveX 对象：

*variable*=new ActiveXObject("Microsoft.XMLHTTP");

为了应对所有的现代浏览器，包括 IE5 和 IE6，请检查浏览器是否支持 XMLHttpRequest 对象。如果支持，则创建 XMLHttpRequest 对象。如果不支持，则创建 ActiveXObject ：

## 向服务器发送请求

如需将请求发送到服务器，我们使用 XMLHttpRequest 对象的 open() 和 send() 方法：

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

|  |  |
| --- | --- |
| **方法** | **描述** |
| open(*method*,*url*,*async*) | 规定请求的类型、URL 以及是否异步处理请求。   * *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） |
| send(*string*) | 将请求发送到服务器。   * *string*：仅用于 POST 请求 |

## GET 还是 POST？

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

## GET 请求

一个简单的 GET 请求：

## 实例

xmlhttp.open("GET","/try/ajax/demo\_get.php",true); xmlhttp.send();

**[尝试一下 »](https://www.runoob.com/try/try.php?filename=tryajax_get" \t "https://www.runoob.com/ajax/_blank)**

在上面的例子中，您可能得到的是缓存的结果。

为了避免这种情况，请向 URL 添加一个唯一的 ID：

## 实例

xmlhttp.open("GET","/try/ajax/demo\_get.php?t=" + Math.random(),true); xmlhttp.send();

**[尝试一下 »](https://www.runoob.com/try/try.php?filename=tryajax_get_unique" \t "https://www.runoob.com/ajax/_blank)**

如果您希望通过 GET 方法发送信息，请向 URL 添加信息：

## 实例

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

**[尝试一下 »](https://www.runoob.com/try/try.php?filename=tryajax_get2" \t "https://www.runoob.com/ajax/_blank)**

## POST 请求

一个简单 POST 请求：

## 实例

xmlhttp.open("POST","/try/ajax/demo\_post.php",true); xmlhttp.send();

**[尝试一下 »](https://www.runoob.com/try/try.php?filename=tryajax_post" \t "https://www.runoob.com/ajax/_blank)**

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：

## 实例

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

**[尝试一下 »](https://www.runoob.com/try/try.php?filename=tryajax_post2" \t "https://www.runoob.com/ajax/_blank)**

|  |  |
| --- | --- |
| **方法** | **描述** |
| setRequestHeader(*header,value*) | 向请求添加 HTTP 头。   * *header*: 规定头的名称 * *value*: 规定头的值 |

## url - 服务器上的文件

open() 方法的 *url* 参数是服务器上文件的地址：

xmlhttp.open("GET","ajax\_test.html",true);

该文件可以是任何类型的文件，比如 .txt 和 .xml，或者服务器脚本文件，比如 .asp 和 .php （在传回响应之前，能够在服务器上执行任务）。

## 异步 - True 或 False？

AJAX 指的是异步 JavaScript 和 XML（Asynchronous JavaScript and XML）。

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：

xmlhttp.open("GET","ajax\_test.html",true);

对于 web 开发人员来说，发送异步请求是一个巨大的进步。很多在服务器执行的任务都相当费时。AJAX 出现之前，这可能会引起应用程序挂起或停止。

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=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：

## 实例

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();

**[尝试一下 »](https://www.runoob.com/try/try.php?filename=tryajax_first" \t "https://www.runoob.com/ajax/_blank)**

您将在稍后的章节学习更多有关 onreadystatechange 的内容。

## Async = false

如需使用 async=false，请将 open() 方法中的第三个参数改为 false：

xmlhttp.open("GET","test1.txt",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：

## 实例

xmlhttp.open("GET","/try/ajax/ajax\_info.txt",false); xmlhttp.send(); document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

# AJAX - 服务器 响应

## 服务器响应

如需获得来自服务器的响应，请使用 XMLHttpRequest 对象的 responseText 或 responseXML 属性。

|  |  |
| --- | --- |
| **属性** | **描述** |
| responseText | 获得字符串形式的响应数据。 |
| responseXML | 获得 XML 形式的响应数据。 |

## responseText 属性

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：

## 实例

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

**[尝试一下 »](https://www.runoob.com/try/try.php?filename=tryajax_first" \t "https://www.runoob.com/ajax/_blank)**

## responseXML 属性

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：

# AJAX - onreadystatechange 事件

## onreadystatechange 事件

当请求被发送到服务器时，我们需要执行一些基于响应的任务。

每当 readyState 改变时，就会触发 onreadystatechange 事件。

readyState 属性存有 XMLHttpRequest 的状态信息。

下面是 XMLHttpRequest 对象的三个重要的属性：

|  |  |
| --- | --- |
| **属性** | **描述** |
| 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 |

在 onreadystatechange 事件中，我们规定当服务器响应已做好被处理的准备时所执行的任务。

当 readyState 等于 4 且状态为 200 时，表示响应已就绪：

## 使用回调函数

回调函数是一种以参数形式传递给另一个函数的函数。

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。

该函数调用应该包含 URL 以及发生 onreadystatechange 事件时执行的任务（每次调用可能不尽相同）：

服务器常用的状态码及其对应的含义如下：

* + **200**：服务器响应正常。
  + **304**：该资源在上次请求之后没有任何修改（这通常用于浏览器的缓存机制，使用GET请求时尤其需要注意）。
  + **400**：无法找到请求的资源。
  + **401**：访问资源的权限不够。
  + **403**：没有权限访问资源。
  + **404**：需要访问的资源不存在。
  + **405**：需要访问的资源被禁止。
  + **407**：访问的资源需要代理身份验证。
  + **414**：请求的URL太长。
  + **500**：服务器内部错误。

[CBH](https://www.runoob.com/ajax/javascript:;" \t "https://www.runoob.com/ajax/_blank)

   CBH

  155\*\*\*73125@163.com

2年前 (2017-04-11)

   1778757629

  lix\*\*\*778757629@gmail.com

补全状态码避免再去搜:

100——客户必须继续发出请求

101——客户要求服务器根据请求转换HTTP协议版本

200——交易成功

201——提示知道新文件的URL

202——接受和处理、但处理未完成

203——返回信息不确定或不完整

204——请求收到，但返回信息为空

205——服务器完成了请求，用户代理必须复位当前已经浏览过的文件

206——服务器已经完成了部分用户的GET请求

300——请求的资源可在多处得到

301——删除请求数据

302——在其他地址发现了请求数据

303——建议客户访问其他URL或访问方式

304——客户端已经执行了GET，但文件未变化

305——请求的资源必须从服务器指定的地址得到

306——前一版本HTTP中使用的代码，现行版本中不再使用

307——申明请求的资源临时性删除

400——错误请求，如语法错误

401——请求授权失败

402——保留有效ChargeTo头响应

403——请求不允许

404——没有发现文件、查询或URl

405——用户在Request-Line字段定义的方法不允许

406——根据用户发送的Accept拖，请求资源不可访问

407——类似401，用户必须首先在代理服务器上得到授权

408——客户端没有在用户指定的时间内完成请求

409——对当前资源状态，请求不能完成

410——服务器上不再有此资源且无进一步的参考地址

411——服务器拒绝用户定义的Content-Length属性请求

412——一个或多个请求头字段在当前请求中错误

413——请求的资源大于服务器允许的大小

414——请求的资源URL长于服务器允许的长度

415——请求资源不支持请求项目格式

416——请求中包含Range请求头字段，在当前请求资源范围内没有range指示值，请求也不包含If-Range请求头字段

417——服务器不满足请求Expect头字段指定的期望值，如果是代理服务器，可能是下一级服务器不能满足请求

500——服务器产生内部错误

501——服务器不支持请求的函数

502——服务器暂时不可用，有时是为了防止发生系统过载

503——服务器过载或暂停维修

504——关口过载，服务器使用另一个关口或服务来响应用户，等待时间设定值较长

505——服务器不支持或拒绝支请求头中指定的HTTP版本

[1778757629](https://www.runoob.com/ajax/javascript:;" \t "https://www.runoob.com/ajax/_blank)

   1778757629

  lix\*\*\*778757629@gmail.com

2年前 (2017-08-21)

   CatnapSui

  sui\*\*\*nan163@163.com

[参考地址](https://www.runoob.com/w3cnote/onreadystatechange-attribute-learn.html" \t "https://www.runoob.com/ajax/_blank)

用 **window.alert** 跟踪 **xmlHttp.readystate** 的变化，发现于原来它运行的机制是这样的。首先创建一个 **xmlHttpRequest** 的对象之后 **xmlHttp.readyState** 的值是 **0** 了，然后**xmlHttp.onreadystatechange = handlestatechange** 没有运行。紧接着是 **open()**，这个函数发生了之后 **xmlHttp.readyState** 的值是 **1** 了，那么就会有一个断点在 **Open()** 函数处断开，保留现场，紧接着又返回到 **xmlHttp.onreadystatechange = handlestatechange** 运行，然后再执行 **Send()** 函数，这个函数发生了之后 **xmlHttp.readyState** 的值是 **2** 了，接着又返回到 **xmlHttp.onreadystatechange = handlestatechange** 运行，以此类推。

# AJAX ASP/PHP 实例

AJAX 用于创造动态性更强的应用程序。

## AJAX ASP/PHP 实例

下面的例子将为您演示当用户在输入框中键入字符时，网页如何与 web 服务器进行通信： 请在下面的输入框中键入字母（A - Z）：