AJAX 可以用于创建快速动态的网页。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。

要创建AJAX实例，需要使用服务器端语言，例如Servlet，JSP，PHP，的ASP.Net。以JSP为例，创建AJAX的实例步骤：

1. 加载org.json.jar文件
2. 创建输入页面以接收任何文本或数字
3. 创建服务器端页面以处理请求
4. 在web.xml文件中提供条目

XHR 创建对象

XMLHttpRequest 对象，可扩展超文本传输请求。用于客户端和服务器之间的异步通信。它执行以下操作：

1. 在后台从客户端发送数据
2. 从服务器接收数据
3. 更新网页而不重新加载。

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");
```

XMLHttpRequest对象的常见属性如下：

属性	描述
onreadystatechange	存储函数（或函数名），每当readyState的属性改变时，就会调用该函数。 注意：`onreadystatechange`事件被触发 5 次（0 - 4），对应着`readyState`的每个变化。
readyState	存有的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
reponseText	以文本形式返回响应。
responseXML	以XML格式返回响应
status	将状态返回为数字（例如，“Not Found”为404，“OK”为200）
statusText	以字符串形式返回状态（例如，“Not Found”或“OK”）

readyState状态说明

0：请求未初始化

此阶段确认XMLHttpRequest对象是否创建，并为调用`open()`方法为初始化作好准备。值为0表示对象已经存在，否则浏览器会报错：对象不存在。

1：服务器连接已建立

此阶段对XMLHttpRequest对象进行初始化，即调用`open()`方法，根据参数`（methoa, url, true）`完成对象状态的设置。并调用`send()`方法开始向服务端发送请求。值为1表示正在向服务端发送请求。

2：请求已接收

此阶段接收服务器端的响应数据。但获得的还只是服务端响应的原始数据，并不能直接在客户端使用。值为2表示已经接收完全部响应数据，并为下一阶段对数据解析作好准备。

3：请求处理中

此阶段解析接收到的服务器端响应数据即根据服务器端响应头部返回的MIME类型把数据转换成能通过`responseBody`，`responseText`或`responseXML`的属性存取的格式，为在客户端调用作好准备。状态3表示正在解析数据。

4：请求已完成，且响应已就绪

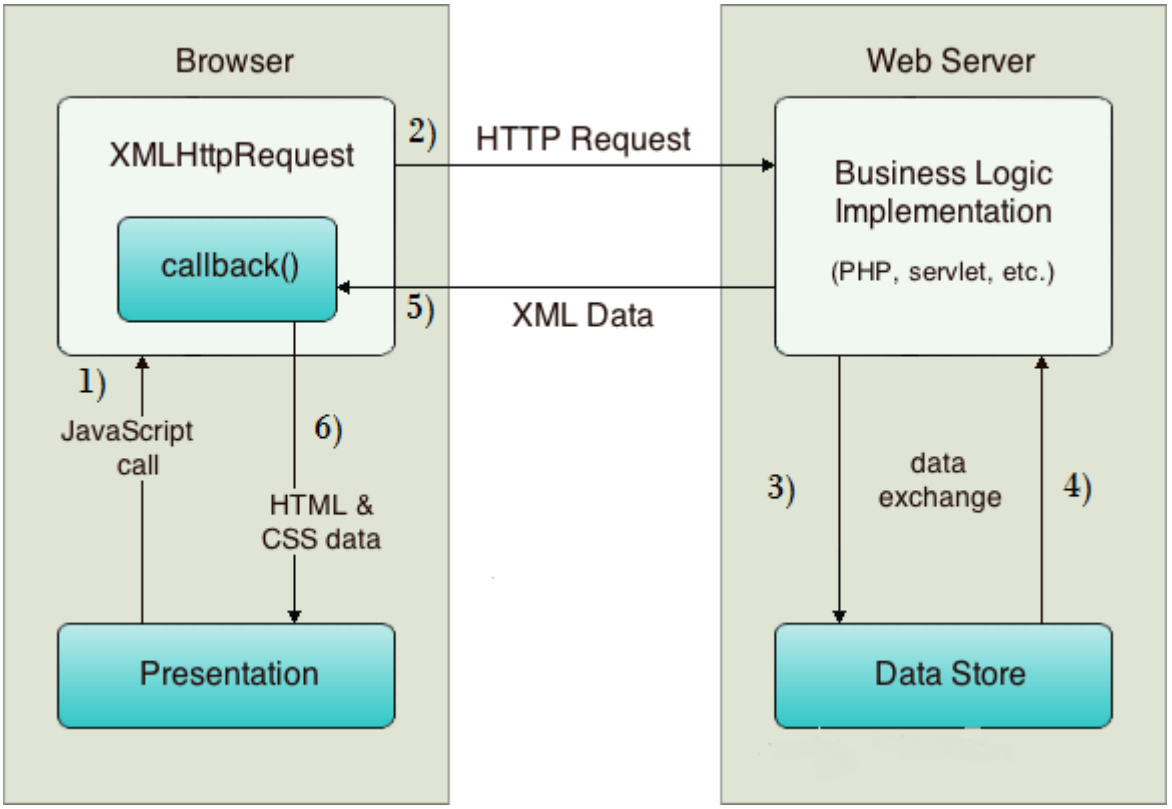
此阶段确认全部数据都已经解析为客户端可用的格式，解析已经完成。值为4表示数据解析完毕，可以通过的XMLHttpRequest对象的属性

取得数据。

XMLHttpRequest对象的重要方法如下：

方法	描述
abort()	取消当前请求。
getAllResponseHeaders()	以字符串形式返回完整的HTTP标头集。
getResponseHeader(headerName)	返回指定HTTP标头的值。
void open (method，URL)	打开指定获取或交的方法和URL的请求。
void open (method，URL，async)	与上面相同，但指定异步或不。
void open (method，URL，async，userName，password)	与上面相同，但指定用户名和密码。
void send (content)	发送获取请求。
setRequestHeader (header，value)	将标签/值对添加到要发送的HTTP标头。 *header*: 规定头的名称 *value*: 规定头的值

XHR 请求



1. 用户从UI发送请求，JavaScript中调用XMLHttpRequest对象。
2. HTTP请求由XMLHttpRequest对象发送到服务器。
3. 服务器使用JSP，PHP，Servlet，ASP.net等与数据库交互。
4. 检索数据。
5. 服务器将XML数据或JSON数据发送到XMLHttpRequest回调函数。
6. HTML和CSS数据显示在浏览器上。

XMLHttpRequest对象的生命周期应该包含如下阶段：

1. 创建
2. 初始化请求
3. 发送请求
4. 接收数据
5. 解析数据
6. 完成

向服务器发送请求，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,可以是任何类型。 *async*：true（异步）或 false（同步）用于AJAX必须异步
send(*string*)	将请求发送到服务器。 *string*：仅用于 POST 请求

与`POST`相比，GET 更简单也更快，并且在大部分情况下都能用。但在以下情况中，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

- GET请求可被缓存
- GET请求保留在浏览器历史记录中
- GET请求可被收藏为书签
- GET请求不应在处理敏感数据时使用
- GET请求有长度限制
- GET请求只应当用于取回数据

POST:

- POST请求不会被缓存
- POST请求不会保留在浏览器历史记录中
- POST请求不能被收藏为书签
- POST请求对数据长度没有要求

像 HTML 表单那样 POST 数据，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","ajax_test.html",true);
xmlhttp.setRequestHeader("Content-
type","application/x-www-form-urlencoded");
xmlhttp.send("fname=Henry&lname=Ford");
```

XHR 响应

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

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。

```
eg.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;
```

xml文档

```
<CATALOG>
<CD>
<TITLE>Empire Burlesque</TITLE>
<ARTIST>Bob Dylan</ARTIST>
<COUNTRY>USA</COUNTRY>
<COMPANY>Columbia</COMPANY>
<PRICE>10.90</PRICE>
<YEAR>1985</YEAR>
</CD>
</CATALOG>
```

回调函数

如果您的网站上存在多个 AJAX 任务，那么应该为创建 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

实例

```
function loadXMLDoc(url,cfunc)
{
```

```
    if (window.XMLHttpRequest)
      {// IE7+, Firefox, Chrome, Opera, Safari 代码
          xmlhttp=new XMLHttpRequest();
      }
    else
      {// IE6, IE5 代码
          xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
      }
    xmlhttp.onreadystatechange=cfunc;
    xmlhttp.open("GET",url,true);
    xmlhttp.send();
}
function myFunction()
{
    loadXMLDoc("ajax_info.txt",function()
       {
           if (xmlhttp.readyState==4 && xmlhttp.status==200)
                {

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