**2.1 Ajax简介**

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。

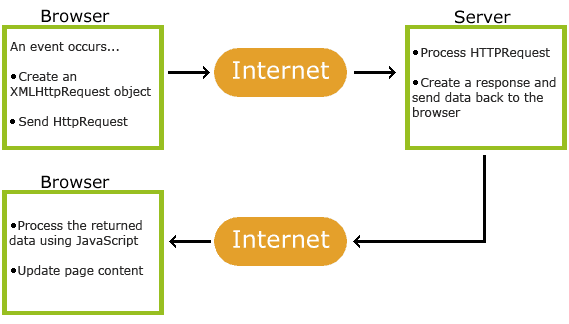
Ajax 最大的优点是在不重新加载整个页面的情况下，可以与服务器交换数据并更新部分网页内容。

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在2005年由Jesse James Garett首先提也。在接下来的极短的时间里，ajax被广泛应用到B/S的应用中，改进了传统的web应用，给浏览者一种更连续的体验。ajax最大的优势在于异步交互，即浏览者在浏览页面时，可同时向服务器发送请求，甚至可以不用等待前一次请求得到完全响应，便再次发送请求。

使用ajax异步请求方式，浏览器无须频繁地得重新加载新页面，服务器的响应不再是整个页面内容，而只是必须更新的部分数据。ajax可以减轻服务器和带宽的负担，提供更好的服务响应。

**2.2 Ajax原理**



**2.3 Ajax - 创建 XMLHttpRequest 对象**

XMLHttpRequest 是 Ajax 的基础。

所有现代浏览器（IE7+、Firefox、Chrome、Safari 以及 Opera） 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 .

**2.4 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.

|  |  |
| --- | --- |
| **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 |

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。

|  |  |
| --- | --- |
| **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 |

xmlhttp.setRequestHeader("Content-Type","application/x-www-form-urlencoded"); xmlhttp.send("fname=Henry&lname=Ford");

**2.5 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：

**2.6 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.

**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.

**2.7 跨域的问题**

域： 协议+域名+端口

跨域请求（访问）：一个域下的文件请求另外一个域下的资源就产生了跨域。（禁止访问）

**跨域的解决之案**

通过服务端代理（不属于跨域）服务器之间是不存在跨域的

Jsonp: json with padding是采用callback赋值函数来获取对象

例如：https://api.douban.com/v2/book/search?q=0&callback=aa

注意：

使用原生和jq的ajax方式都可能会存在跨域问题，解决方式？

1. 自己写服务器中转。
2. 不使用ajax，采用jsonp的方式访问后端可不使用ajax。
3. 原生jsonp，（自己定义约定的回调函数）。
4. jQuery的jsonp（将回调函数的内容写在success方法里，succe方法相当于自己定义的回调函数）。当$.ajax({…})中的dataType设置为jsonp时，其实已经和ajax无关了，是使用jsonp的方式实现请获取数据的，其原理同原生jsonp。

# 跨域

var xhr = new XMLHttpRequest();

console.log(xhr);

// 3.2） 发送请求 open, send

xhr.open("get", url, true);

// 3.3) 监听状态的改变事件

xhr.onreadystatechange = function(){

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

// 3.4) 获取响应数据

/\*

\* bookList.html?\_ijt=nkl5bl4vibdeib39lf5i83o0mu:1 XMLHttpRequest cannot load

\* https://api.douban.com/v2/book/search?q=ff. No 'Access-Control-Allow-Origin' header

\* is present on the requested resource.

\* Origin 'http://localhost:63342' is therefore not allowed access.

\*

\* 以上错误是由跨域引起的。

\* 什么是跨域？

\* http://domain-name:port (协议 + 域名 + 端口）

\* 协议 + 域名 + 端口 中只要有任何一个不同，即为跨域

\*

\* 前端： http://localhost:63342

\* 后端： https://api.douban.com

\*

\* 解决办法一： 自己写服务器中转

\* 前端： http://localhost:63342

\* 后端： http://localhost:8081 （依然跨域，但是我们可以在后端代码中设置Access-Control-Allow-Origin' header

\*

\* 修改原url地址为： http://192.168.2.54:8081?myUrl=https://api.douban.com/v2/book/search?q=

\* \*/

var data = xhr.responseText; //得到的是json格式的字符串

data = JSON.parse(data);

异步与同步

使用setTimeout实现异步函数，事件不同时进行，先完成某件后再做某个，

同步，按顺序执行事件。

XMLHttpRequest是Ajax的基础，ie7之下使用ActiveXOvject来代替，其他的版本或浏览器均支持。

异步，事件正常会依次执行，当上一个执行完毕执行下一个，若到了异步代码会将异步代码放到事件队列中，在事件队列中对异步代码进行依次排队。当异步进行完毕会发送通知，然后将返回数据显示出来。

Form表单提交无刷新

<iframe name="targetIfr" style="display:none"></iframe>

<form action="http://api.yunqihui.net/chuizi\_wchat/system/generalizeOrder/save/json" target="targetIfr" onsubmit="checkUser()">

</form>