**Ajax**

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。

**什么是Ajax**

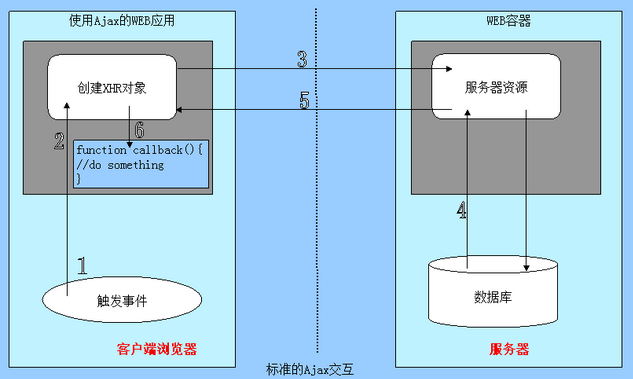
AJAX = 异步 JavaScript 和 XML。

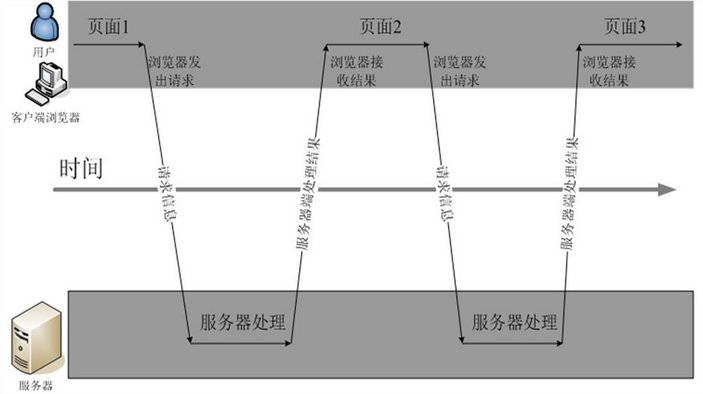
AJAX 是一种用于创建快速动态网页的技术。

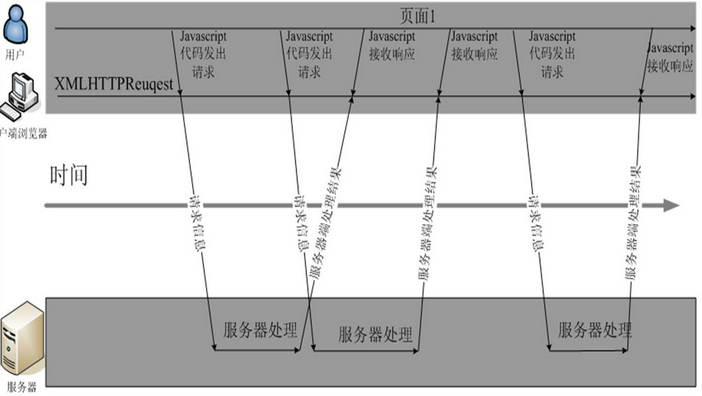
通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

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。

**Ajax交互模型**

[](https://cloud.githubusercontent.com/assets/14271179/14113417/7507c986-f605-11e5-8221-a4678d467027.png)

浏览器的普通交互方式  
[](https://cloud.githubusercontent.com/assets/14271179/14113405/665f4f9e-f605-11e5-900e-98d58a8294aa.png)

浏览器的Ajax交互方式  
[](https://cloud.githubusercontent.com/assets/14271179/14113433/7e104b2a-f605-11e5-98f1-712cbf08935e.png)

**Ajax应用**

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。

**XMLHttpRequest**

XMLHttpRequest是ajax中的一个最为重要的对象，可以说没有xmlHttpRequst的支持就没有ajax技术。

1. 创建对象
2. xmlRequest = new XMLHttpRequest(); //IE7 & Mozilla

xmlRequest = new ActiveXObject("Microsoft.XMLHTTP"); //IE6,IE5

1. xmlhttprequest 具有六个常用的方法，分别是：
   * abort---------停止当前的请求.
   * getAllResponseHeader-----------把当前的所有HTTP请求的响应头部作为键值对返回.
   * getResponseHeader(“header”)----------返回指定的首部串值.
   * open(“method”,”url”,”Boolean”,”username”,”password”)------建立对服务器的请求,其中method方法为get,put,post等.
   * send(content)-------------------------具休向服务器提出请求.
   * setRequestHeader(“header”,”value”)-----------为指定的首部设置值,注在设置之前,必须先调用OPEN()方法.
2. xmlhttprequest还提供了六个常用的属性.
   * onreadystatechange--------------------当状态改变时就调用由此指定的函数.
   * readystate----------------------------获得当前状态.通常有5个可取值(0-4)对我们真真正用用的就是4代表完成. .
   * responseText--------------------------通常由服务器返回一个字符串.
   * responseXML---------------------------表示返回一个XML对象.
   * status--------------------------------返回服务器状态.(200代表OK,404代表没有找到)
   * statuse-------------------------------http状态码的相应文本.(ok或未找到).
3. readystate属性
   * 0 － （未初始化）还没有调用send()方法
   * 1 － （载入）已调用send()方法，正在发送请求
   * 2 － （载入完成）send()方法执行完成，已经接收到全部响应内容
   * 3 － （交互）正在解析响应内容
   * 4 － （完成）响应内容解析完成，可以在客户端调用了

**Ajax实现**

**Ajax实现分为四部分：**

1. 创建Ajax对象
2. // 创建ajax对象var xhr = null;
3. if(window.XMLHttpRequest){
4. xhr = new XMLHttpRequest();
5. } else {
6. //为了兼容IE6
7. xhr = new ActiveXObject('Microsoft.XMLHTTP');

}

1. 连接服务器
2. // 连接服务器open(方法GET/POST，请求地址， 异步传输)

xhr.open('GET', 'data.txt', true);

1. 发送请求
2. // 发送请求

xhr.send();

1. 接收返回数据
2. // 处理返回数据
3. /\*
4. \*\* 每当readyState改变时，就会触发onreadystatechange事件
5. \*\* readyState属性存储有XMLHttpRequest的状态信息
6. \*\* 0 ：请求未初始化
7. \*\* 1 ：服务器连接已建立
8. \*\* 2 ：请求已接受
9. \*\* 3 : 请求处理中
10. \*\* 4 ：请求已完成，且相应就绪
11. \*/
12. xhr.onreadystatechange = function(){
13. if(xhr.readyState == 4){
14. /\*
15. \*\* Http状态码
16. \*\* 1xx ：信息展示
17. \*\* 2xx ：成功
18. \*\* 3xx ：重定向
19. \*\* 4xx : 客户端错误
20. \*\* 5xx ：服务器端错误
21. \*/
22. if(xhr.status == 200){
23. success(xhr.responseText);
24. } else {
25. if(failed){
26. failed(xhr.status);
27. }
28. }
29. }

}

**Ajax封装函数**

function Ajax(type, url, data, success, failed){

// 创建ajax对象

var xhr = null;

if(window.XMLHttpRequest){

xhr = new XMLHttpRequest();

} else {

xhr = new ActiveXObject('Microsoft.XMLHTTP')

}

var type = type.toUpperCase();

// 用于清除缓存

var random = Math.random();

if(typeof data == 'object'){

var str = '';

for(var key in data){

str += key+'='+data[key]+'&';

}

data = str.replace(/&$/, '');

}

if(type == 'GET'){

if(data){

xhr.open('GET', url + '?' + data, true);

} else {

xhr.open('GET', url + '?t=' + random, true);

}

xhr.send();

} else if(type == 'POST'){

xhr.open('POST', url, true);

// 如果需要像 html 表单那样 POST 数据，请使用 setRequestHeader() 来添加 http 头。

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

xhr.send(data);

}

// 处理返回数据

xhr.onreadystatechange = function(){

if(xhr.readyState == 4){

if(xhr.status == 200){

success(xhr.responseText);

} else {

if(failed){

failed(xhr.status);

}

}

}

}

}

// 测试调用

var sendData = {name:'asher',sex:'male'};

Ajax('get', 'data/data.html', sendData, function(data){

console.log(data);

}, function(error){

console.log(error);

});

**Ajax优缺点**

**Ajax优点**

1. Ajax实现动态不刷新（局部刷新）
2. 通过异步模式，提升了用户体验
3. 优化了浏览器和服务器之间的传输，减少不必要的数据往返，减少了带宽占用
4. Ajax在客户端运行，承担了一部分本来由服务器承担的工作，减少了大用户量下的服务器负载。

**Ajax缺点**

1. 以无刷新改变页面内容，但无法改变页面URL
2. 为了更好的可访问性，内容发生改变后，通常改变URL的hash
3. hash的方式不能很好的处理浏览器的前进、后退等问题
4. 进而浏览器引入了onhashchange的接口，不支持的浏览器只能定时去判断hash是否改变
5. 但这种方式对搜索引擎很不友好
6. twitter和google约定了使用#!xxx（即hash第一个字符为!），搜索引擎进行支持。

**参考文献**

JavaScript学习总结（七）Ajax和Http状态字  
<https://segmentfault.com/a/1190000000691919>