# Ajax

## 背景

1. 传统的Web网站，提交表单，需要重新加载整个页面。
2. **在服务器为响应完成，客户端不能做任何操作。**

     2.如果服务器长时间未能返回Response，则客户端将会无响应，用户体验很差。

     3.服务端返回Response后，浏览器需要加载整个页面，对浏览器的负担也是很大的。

     4.浏览器提交表单后，发送的数据量大，造成网络的性能问题。

## 什么是AJAX

AJAX全称为“Asynchronous JavaScript and XML”（异步JavaScript和XML），是一种创建交互式网页应用的网页开发技术。

ajax并非一种新的技术，而是几种原有技术的结合体。它由下列技术组合而成。

   1.使用CSS和XHTML来表示。

   2.使用DOM模型来交互和动态显示。

   3.使用XMLHttpRequest来和服务器进行异步通信。

   4.使用javascript来绑定和调用。

AJAX的概念：

1.AJAX = 异步 JavaScript 和 XML。

2.AJAX 是一种用于创建快速动态网页的技术。

3.通过在后台与服务器进行少量数据交换，可以使网页实现异步更新。

4.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。

## AJAX的工作原理

Ajax的工作原理相当于在用户和服务器之间加了—个中间层(AJAX引擎),使用户操作与服务器响应异步化。并不是所有的用户请求都提交给服务器,像—些数据验证和数据处理等都交给Ajax引擎自己来做, 只有确定需要从服务器读取新数据时再由Ajax引擎代为向服务器提交请求。

Ajax的原理简单来说通过XmlHttpRequest对象来向服务器发异步请求，从服务器获得数据，然后用javascript来操作DOM而更新页面。

**什么叫异步**

当前页面发送一个请求给服务器，**当前页面不需要等待服务器响应才能操作网页。发送完请求之后，当前页面可以继续浏览，操作。**

**为什么需要AJAX**

当需要从服务器获取数据，并刷新页面的操作，如果不采用AJAX，则需要用提交整个表单的方式，当提交表单时，发送请求给服务器，页面需要等待服务器发送完response后，页面才能恢复操作。

## JS原生ajax

创建异步对象 var ajax = new XMLHTTPRequest()

设置请求行ajax.open('方法','地址');

设置请求头ajax.setRequestHeader('content-type','application/x-www-form-urlencoded')

注册回调函数ajax.onload = function(){}

普通的字符串ajax.responseText

XML ajax.responseXML

JSON JSON.parse(ajax.responseText)

设置请求主体ajax.send('key=value&key2=vlaue2')

### 创建Ajax核心对象XMLHttpRequest(记得考虑兼容性)

1. var xhr=null;

2. if (window.XMLHttpRequest)

3. {// 兼容 IE7+, Firefox, Chrome, Opera, Safari

4. xhr=new XMLHttpRequest();

5. } else{// 兼容 IE6, IE5

6. xhr=new ActiveXObject("Microsoft.XMLHTTP");

7. }

### post请求一定要设置请求头的格式内容

ajax.setRequestHeader('content-type','application/x-www-form-urlencoded')

### ****get\post区别：****

****

****

### ****XMLHttpRequest 对象的知识****

  　　它的属性有：

  　　onreadystatechange  每次状态改变所触发事件的事件处理程序。

  　　responseText     从服务器进程返回数据的字符串形式。

  　　responseXML    从服务器进程返回的DOM兼容的文档数据对象。

  　　status           从服务器返回的数字代码，比如常见的404（未找到）和200（已就绪）

  　　status Text       伴随状态码的字符串信息

  　　readyState       对象状态值

　　　　0 (未初始化) 对象已建立，但是尚未初始化（尚未调用open方法）

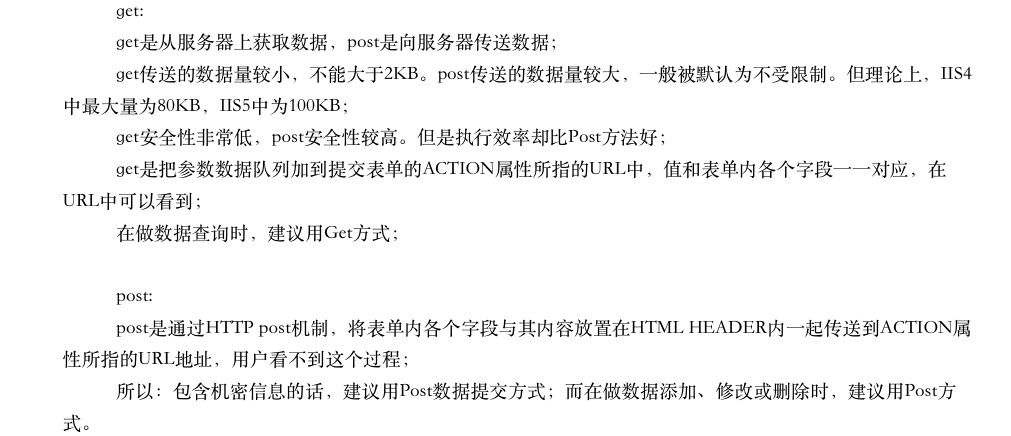
　　　　1 (初始化) 对象已建立，尚未调用send方法

　　　　2 (发送数据) send方法已调用，但是当前的状态及http头未知

　　　　3 (数据传送中) 已接收部分数据，因为响应及http头不全，这时通过responseBody和responseText获取部分数据会出现错误，

　　　　4 (完成) 数据接收完毕,此时可以通过通过responseXml和responseText获取完整的回应数据

### get和post两种方式的优缺点



### HTTP协议

请求报文

请求行

请求头

请求主体

响应报文

状态行

响应头

响应主体

**什么叫局部刷新**

**第一种：iframe页面重载的方式**

这种方式虽然实现了部分刷新，但是是页面的重载，所以也会带来性能上的问题。

Step1.在页面中定义一个Iframe

[?](https://www.jb51.net/article/90528.htm)

|  |  |
| --- | --- |
| 1  2 | <iframe id="indexFrame" name="index" width="1000" height="800"      frameborder="0" marginwidth="0" marginheight="0" scrolling="yes" style="margin-top:100px;"></iframe> |

**Step2.设置Iframe的src**

[?](https://www.jb51.net/article/90528.htm)

|  |  |
| --- | --- |
| 1  2 | var indexFrame = document.getElementById("indexFrame");  indexFrame.src = "introduction.php"; |

**Step3.添加一个button的点击事件**，当点击这个button时，重新设置Iframe的src，实现iframe里面的页面刷新。Iframe外面的内容不刷新。

[?](https://www.jb51.net/article/90528.htm)

|  |  |
| --- | --- |
| 1 | <button id="room" onclick='IndexClick("room")'>Click Me!</button> |

[?](https://www.jb51.net/article/90528.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | function IndexClick(moduleKey) {   var indexFrame = document.getElementById("indexFrame");   if(indexFrame == null)   {     indexFrame = parent.document.getElementById("indexFrame");   }   var url = "introduction.php";   switch (moduleKey) {    case "introduction":     url = "introduction.php";     break;    case "room":     url = "room.php";     break;    default:     {     }   }   indexFrame.src = url;  } |

通过这种方式我们可以实现一个导航栏的功能：

**第二种：AJAX方式**

Step1.JavaScrpit发送异步请求

Step2.服务端查询数据库，返回数据

Step3.服务端返回Response

Step4.客户端根据返回的Response，来用JavaScript操作DOM。

**AJAX  GET和POST方式区别**

## ****JQuery实现AJAX****

function getWeeklyCalendar(name,currentDate,mode){

 $.ajax({

   type:'POST',

<url:'weekProcess.php>',

data:'func=getWeeklyCalender&name='+name+'¤tDate='+currentDate+'& mode='+mode,

   success:function(data){

   console.log(data);

   }

 });

}

## ****跨域(额外)****

(1)jsonp是用script标签的src属性是不跨域的这一性质，所以其实是封装了这个功能而已，jquery会创建一个script标签，把src的地址指向后端，src会带一个callback参数，一般是一个函数名，后端根据这个请求，获取参数，然后把需要返回的数据包裹在这个函数内，前端获得了这些js代码，就会执行这个callback，自然就把数据传到客户端了。只支持GET请求参考文档

(2)cors(cross-origin resource sharing)就是服务端加上一句 header(“Access-Control-Allow-Origin:\*”); 支持所有请求，但是兼容性不太好，支持IE8+,chrome3+。

(3)window.name.一个窗口window的生命周期内，窗口载入的页面共享一个window.name,每个页面都有读写权限, data.html里面就写上window.name=”data..”;然后在a.html里用一个隐藏的iframe载入data.html,然后在a.html里用js把iframe的src设为同源的一个页面。

(4)修改document.domain的方法只适用于不同子域的框架间的交互。比如http://www.example.com/a.html和 http:example.com/b.html 在两个页面中都修改document.domain为”example.com”

(5)img标签的src也是不跨域的，所以可以

img.src=“http://example.com/data?value=123。

但是这种方法只能用来发送请求.

(6)HTML5有一个postMessage(data,origin)方法，可以向当前页面中的iframe或者当前页弹出的窗口发送消息

## **优点**

     1.使用异步方式与服务器通信，页面不需要重新加载，页面无刷新

     2.按需取数据，减少服务器的负担

     3.使得Web应用程序更为迅捷地响应用户交互

     4.AJAX基于标准化的并被广泛支持的技术，不需要下载浏览器插件或者小程序，但需要客户允许JavaScript在浏览器上执行

     5.浏览器的内容和服务端代码进行分离。页面的内容全部由JAVAScript来控制，服务端负责逻辑的校验和从数据库中拿数据。

## **缺点**

     1.安全问题：将服务端的方法暴露出来，黑客可利用这一点进行攻击

     2.大量JS代码，容易出错

     3.Ajax的无刷新重载，由于页面的变化没有刷新重载那么明显，所以容易给用户带来困扰——用户不太清楚现在的数据是新的还是已经更新过的；现有的解决有：在相关位置提示、数据更新的区域设计得比较明显、数据更新后给用户提示等

     4.可能破坏浏览器后退按钮的正常行为；

     5.一些手持设备（如手机、PAD等）自带的浏览器现在还不能很好的支持Ajax

## **应用场景**

     1.对数据进行过滤和操纵相关数据的场景

     2.添加/删除树节点

     3.添加/删除列表中的某一行记录

     4.切换下拉列表item

     5.注册用户名重名的校验

## **不适用场景**

     1.整个页面内容的保存

     2.导航