ajax数据

Ajax入门

2017 年 01 月 18 日

Damonare

前言

总括： 本文讲解了ajax的历史，工作原理以及优缺点，对XMLHttpRequest对象进行了详细的讲解，并使用原生js实现了一个ajax对象以方便日常开始使用。

- damonare的ajax库：damonare的ajax库
- 原文博客地址：你真的懂ajax吗？
- 知乎专栏&&简书专题：前端进击者（知乎）&&前端进击者（简书）
- 博主博客地址：Damonare的个人博客

古之立大事者，不惟有超世之才，亦必有坚忍不拔之志。

正文

相信每个前端程序员日常工作中都避免不了的工作就是和后端联调，联调自然就避免不了使用ajax，但我相信，不管是使用jquery封装的ajax方法还是使用vue的插件vue-resource的程序员，真正对于ajax有过深入探究的并不多，我们更多的是为了使用而使用，至于它的原理往往因为即使不了解依旧能做出东西而懒得去看，我们都被轮子们惯坏了。根据二八定律，即任何一组东西中，最重要的只占其中一小部分，约20%，其余80%的尽管是多数，却是次要的。因 此，如果想挤进那20%的行列，就要学到一般人学不到的深度，学到一般人学不了的东西。好的，现在我们从头来说一下ajax。

Ajax简介

在上世纪90年代，几乎所有的网站都由HTML页面实现，服务器处理每一个用户请求都需要重新加载网页。形式是怎样的呢？就比如说你在浏览器上登录自己的微博账号，填完了表单，点击登录按钮，一次"完整"的HTTP请求就此触发，服务器发现你的登录密码不对头，立马把网页原原本本的返回给你，在用户看来呢，就是一次重新加载的过程！用户体验极差！而且这个做法浪费了许多带宽，因为在前后两个页面中的大部分HTML码往往是相同的。由于每次应用的沟通都需要向服务器发送请求，应用的回应时间依赖于服务器的回应时间。这导致了用户界面的回应比本机应用慢得多。

到了2005年，google率先在它的应用(诸如google地图、gmail)里使用了ajax技术，这才让这项技术正式风靡开来。

如今它的应用已经十分广泛：

- 运用XHTML+CSS来表达信息；
- 运用JavaScript操作DOM（Document Object Model）来运行动态效果；
- 运用XML和XSLT操作数据；
- 运用XMLHttpRequest或新的Fetch API与网页服务器进行异步数据交换；
- 注意：AJAX与Flash、Silverlight和Java Applet等RIA技术是有区分的。

Ajax工作原理

Ajax的工作原理相当于在用户和服务器之间加了一个中间层(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,像一些数据验证(比如判断用户是否输入了数据)和数据处理(比如判断用户输入数据是否是数字)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。把这些交给了Ajax引擎，用户操作起来也就感觉更加流畅了。

Ajax优缺点

Ajax的优点

1.无刷新更新数据。

AJAX最大优点就是能在不刷新整个页面的前提下与服务器通信维护数据。这使得Web应用程序更为迅捷地响应用户交互，并避免了在网络上发送那些没有改变的信息，减少用户等待时间，带来非常好的用户体验。

2.异步与服务器通信。

AJAX使用异步方式与服务器通信，不需要打断用户的操作，具有更加迅速的响应能力。优化了Browser和Server之间的沟通，减少不必要的数据传输、时间及降低网络上数据流量。

3.前端和后端负载平衡。

AJAX可以把以前一些服务器负担的工作转嫁到客户端，利用客户端闲置的能力来处理，减轻服务器和带宽的负担，节约空间和宽带租用成本。并且减轻服务器的负担，AJAX的原则是“按需取数据”，可以最大程度的减少冗余请求和响应对服务器造成的负担，提升站点性能。

4.基于标准被广泛支持。

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。随着Ajax的成熟，一些简化Ajax使用方法的程序库也相继问世。同样，也出现了另一种辅助程序设计的技术，为那些不支持JavaScript的用户提供替代功能。

5.界面与应用分离。

Ajax使WEB中的界面与应用分离（也可以说是数据与呈现分离），有利于分工合作、减少非技术人员对页面的修改造成的WEB应用程序错误、提高效率、也更加适用于现在的发布系统。

Ajax缺点

1.AjAX干掉了Back和加入收藏书签功能，即对浏览器机制的破坏。

对应用Ajax最主要的批评就是，它可能破坏浏览器的后退与加入收藏书签功能。在动态更新页面的情况下，用户无法回到前一个页面状态，这是因为浏览器仅能记下历史记录中的静态页面。一个被完整读入的页面与一个已经被动态修改过的页面之间的可能差别非常微妙；用户通常都希望单击后退按钮，就能够取消他们的前一次操作，但是在Ajax应用程序中，却无法这样做。不过开发者已想出了种种办法来解决这个问题，HTML5 之前的方法大多是在用户单击后退按钮访问历史记录时，通过创建或使用一个隐藏的IFRAME来重现页面上的变更。（例如，当用户在Google Maps中单击后退时，它在一个隐藏的IFRAME中进行搜索，然后将搜索结果反映到Ajax元素上，以便将应用程序状态恢复到当时的状态）。

关于无法将状态加入收藏或书签的问题，HTML5之前的一种方式是使用URL片断标识符（通常被称为锚点，即URL中#后面的部分）来保持追踪，允许用户回到指定的某个应用程序状态。（许多浏览器允许JavaScript动态更新锚点，这使得Ajax应用程序能够在更新显示内容的同时更新锚点。）HTML5 以后可以直接操作浏览历史，并以字符串形式存储网页状态，将网页加入网页收藏夹或书签时状态会被隐形地保留。上述两个方法也可以同时解决无法后退的问题。

2.AJAX的安全问题。

AJAX技术给用户带来很好的用户体验的同时也对IT企业带来了新的安全威胁，Ajax技术就如同对企业数据建立了一个直接通道。这使得开发者在不经意间会暴露比以前更多的数据和服务器逻辑。Ajax的逻辑可以对客户端的安全扫描技术隐藏起来，允许黑客从远端服务器上建立新的攻击。还有Ajax也难以避免一些已知的安全弱点，诸如跨站点脚步攻击、SQL注入攻击和基于Credentials的安全漏洞等等。

3.因为网络延迟需要给用户提供必要提示

进行Ajax开发时，网络延迟——即用户发出请求到服务器发出响应之间的间隔——需要慎重考虑。如果不给予用户明确的回应，没有恰当的预读数据，或者对XMLHttpRequest的不恰当处理，都会使用户感到厌烦。通常的解决方案是，使用一个可视化的组件来告诉用户系统正在进行后台操作并且正在读取数据和内容。

XMLhttpRequest介绍

Ajax(Asynchronous JavaScript and XML)不是指一种单一的技术，而是有机地利用了一系列相关的技术。虽然其名称包含XML，但实际上数据格式可以由JSON代替，进一步减少数据量，形成所谓的AJAJ。为了使用JavaScript向服务器发出 HTTP 请求，需要一个提供此功能的类的实例。这就是XMLHttpRequest的由来。这样的类最初是在Internet Explorer中作为一个名为XMLHTTP的ActiveX对象引入的。然后，Mozilla，Safari和其他浏览器，实现一个XMLHttpRequest类，支持Microsoft的原始ActiveX对象的方法和属性。同时微软也实现了

XMLHttpRequest。

显而易见XMLHttpRequest类是重中之重了。

XMLhttpRequest属性

onreadystatechange

一个JavaScript函数对象，当readyState属性改变时会调用它。回调函数会在user interface线程中调用。

readyState

HTTP 请求的状态.当一个 XMLHttpRequest 初次创建时，这个属性的值从 0 开始，直到接收到完整的 HTTP 响应，这个值增加到 4。

5 个状态中每一个都有一个相关联的非正式的名称，下表列出了状态、名称和含义：

状态	名称	描述
0	Uninitialized	初始化状态。XMLHttpRequest 对象已创建或已被 abort() 方法重置。
1	Open	open() 方法已调用，但是 send() 方法未调用。请求还没有被发送。
2	Sent	Send() 方法已调用，HTTP 请求已发送到 Web 服务器。未接收到响应。
3	Receiving	所有响应头部都已经接收到。响应体开始接收但未完成。
4	Loaded	HTTP 响应已经完全接收。

readyState 的值不会递减，除非当一个请求在处理过程中的时候调用了 abort() 或 open() 方法。每次这个属性的值增加的时候，都会触发 onreadystatechange 事件句柄。

responseText

目前为止为服务器接收到的响应体（不包括头部），或者如果还没有接收到数据的话，就是空字符串。

如果 readyState 小于 3，这个属性就是一个空字符串。当 readyState 为 3，这个属性返回目前已经接收的响应部分。如果 readyState 为 4，这个属性保存了完整的响应体。

如果响应包含了为响应体指定字符编码的头部，就使用该编码。否则，假定使用 Unicode UTF-8。

responseXML

对请求的响应，解析为 XML 并作为 Document 对象返回。

status

由服务器返回的 HTTP 状态代码，如 200 表示成功，而 404 表示 "Not Found" 错误。当 readyState 小于 3 的时候读取这一属性会导致一个异常。

statusText

这个属性用名称而不是数字指定了请求的 HTTP 的状态代码。也就是说，当状态为 200 的时候它是 "OK"，当

状态为 404 的时候它是 "Not Found"。和 status 属性一样，当 readyState 小于 3 的时候读取这一属性会导致一个异常。

XMLHttpRequest方法

abort()

取消当前响应，关闭连接并且结束任何未决的网络活动。

这个方法把 XMLHttpRequest 对象重置为 readyState 为 0 的状态，并且取消所有未决的网络活动。例如，如果请求用了太长时间，而且响应不再必要的时候，可以调用这个方法。

getAllResponseHeaders()

把 HTTP 响应头部作为未解析的字符串返回。

如果 readyState 小于 3，这个方法返回 null。否则，它返回服务器发送的所有 HTTP 响应的头部。头部作为单个的字符串返回，一行一个头部。每行用换行符 "\r\n" 隔开。

getResponseHeader()

返回指定的 HTTP 响应头部的值。其参数是要返回的 HTTP 响应头部的名称。可以使用任何大小写来制定这个头部名字，和响应头部的比较是不区分大小写的。

该方法的返回值是指定的 HTTP 响应头部的值，如果没有接收到这个头部或者 readyState 小于 3 则为空字符串。如果接收到多个有指定名称的头部，这个头部的值被连接起来并返回，使用逗号和空格分隔开各个头部的值。

open()

初始化一个请求. 该方法用于JavaScript代码中;如果是本地代码, 使用 openRequest())方法代替.

> **注意：** 在一个已经激活的request下（已经调用open()或者openRequest()方法的request）再次调用这个方法相当于调用了abort（）方法。

参数

- method

请求所使用的HTTP方法; 例如 "GET", "POST", "PUT", "DELETE"等. 如果下个参数是非HTTP(S)的URL,则忽略该参数.

- url

该请求所要访问的URL

- async

一个可选的布尔值参数，默认为true,意味着是否执行异步操作，如果值为false,则send()方法不会返回任何东西，直到接受到了服务器的返回数据。如果为值为true，一个对开发者透明的通知会发送到相关的事件监听者。这个值必须是true,如果multipart 属性是true，否则将会出现一个意外。

- user

用户名,可选参数,为授权使用;默认参数为空string.

- password

密码,可选参数,为授权使用;默认参数为空string.

send()

发送 HTTP 请求，使用传递给 open() 方法的参数，以及传递给该方法的可选请求体。

setRequestHeader()

向一个打开但未发送的请求设置或添加一个 HTTP 请求(设置请求头)。

参数

- header

将要被赋值的请求头名称

- value

给指定的请求头赋的值

Ajax原生js实现

下面是使用原生js写的ajax：

```
var ajax = {};
ajax.httpRequest = function () {
    //判断是否支持XMLHttpRequest对象
    if (window.XMLHttpRequest) {
        return new XMLHttpRequest();
    }
    //兼容IE浏览器
    var versions = [
        "MSXML2.XmlHttp.6.0",
        "MSXML2.XmlHttp.5.0",
        "MSXML2.XmlHttp.4.0",
        "MSXML2.XmlHttp.3.0",
        "MSXML2.XmlHttp.2.0",
        "Microsoft.XmlHttp"
    ];
    //定义局部变量xhr,储存IE浏览器的ActiveXObject对象
    var xhr;
    for (var i = 0; i < versions.length; i++) {
        try {
            xhr = new ActiveXObject(versions[i]);
            break;
        } catch (e) {
        }
    }
    return xhr;
};

ajax.send = function (url, callback, method, data, async) {
    //默认异步
    if (async === undefined) {
        async = true;
    }
    var httpRequest = ajax.httpRequest();
    //初始化HTTP请求
    httpRequest.open(method, url, async);
    //onreadystatechange函数对象
    httpRequest.onreadystatechange = function () {
        //readyState 的值等于4，从服务器拿到了数据
```

```
        if (httpRequest.readyState == 4) {
            //回调服务器响应数据
            callback(httpRequest.responseText)
        }
    };
    if (method == 'POST') {
          //给指定的HTTP请求头赋值
        httpRequest.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
    }
      //发送HTTP请求
    httpRequest.send(data);
};
//实现GET请求
ajax.get = function (url, data, callback, async) {
    var query = [];
    for (var key in data) {
        query.push(encodeURIComponent(key) + '=' + encodeURIComponent(data[key]));
    }
    ajax.send(url + (query.length ? '?' + query.join('&') : ''), callback, 'GET', null,
async)
};
//实现POST请求
ajax.post = function (url, data, callback, async) {
    var query = [];
    for (var key in data) {
        query.push(encodeURIComponent(key) + '=' + encodeURIComponent(data[key]));
    }
    ajax.send(url, callback, 'POST', query.join('&'), async)
};
```

如果你使用jquery或是zepto很大部分是因为它的ajax兼容性高的缘故，不妨试试这个：[damonare的ajax库]，喜欢给个star也是可以的。