

# XMLHttpRequest Level 2 使用指南

作者：阮一峰

分享

日期：2012年9月 8日



本站由 珠峰培训（专业前端培训）独家赞助

[XMLHttpRequest](#)是一个浏览器接口，使得Javascript可以进行HTTP(S)通信。

最早，微软在IE 5引进了这个接口。因为它太有用，其他浏览器也模仿部署了，ajax操作因此得以诞生。

但是，这个接口一直没有标准化，每家浏览器的实现或多或少有点不同。HTML 5的概念形成后，W3C开始考虑标准化这个接口。2008年2月，就提出了[XMLHttpRequest Level 2](#) 草案。

这个XMLHttpRequest的新版本，提出了很多有用的新功能，将大大推动互联网革新。本文就对这个新版本进行详细介绍。



## 一、老版本的XMLHttpRequest对象

在介绍新版本之前，我们先回顾一下老版本的使用。

首先，新建一个XMLHttpRequest的实例。

```
var xhr = new XMLHttpRequest();
```

然后，向远程主机发出一个HTTP请求。

```
xhr.open('GET', 'example.php');  
  
xhr.send();
```

接着，就等待远程主机做出回应。这时需要监控XMLHttpRequest对象的状态变化，指定回调函数。

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

```
        alert( xhr.responseText );

    } else {

        alert( xhr.statusText );

    }

};
```

上面的代码包含了老版本XMLHttpRequest对象的主要属性：

- \* `xhr.readyState`：XMLHttpRequest对象的状态，等于4表示数据已经接收完毕。
- \* `xhr.status`：服务器返回的状态码，等于200表示一切正常。
- \* `xhr.responseText`：服务器返回的文本数据
- \* `xhr.responseXML`：服务器返回的XML格式的数据
- \* `xhr.statusText`：服务器返回的状态文本。

## 二、老版本的缺点

老版本的XMLHttpRequest对象有以下几个缺点：

- \* 只支持文本数据的传送，无法用来读取和上传二进制文件。
- \* 传送和接收数据时，没有进度信息，只能提示有没有完成。
- \* 受到["同域限制"](#) ( Same Origin Policy )，只能向同一域名的服务器请求数据。

## 三、新版本的功能

新版本的XMLHttpRequest对象，针对老版本的缺点，做出了大幅改进。

- \* 可以设置HTTP请求的时限。
- \* 可以使用FormData对象管理表单数据。

- \* 可以上传文件。
- \* 可以请求不同域名下的数据（跨域请求）。
- \* 可以获取服务器端的二进制数据。
- \* 可以获得数据传输的进度信息。

下面，我就——介绍这些新功能。

#### 四、HTTP请求的时限

有时，ajax操作很耗时，而且无法预知要花多少时间。如果网速很慢，用户可能要等很久。

新版本的XMLHttpRequest对象，增加了timeout属性，可以设置HTTP请求的时限。

```
xhr.timeout = 3000;
```

上面的语句，将最长等待时间设为3000毫秒。过了这个时限，就自动停止HTTP请求。与之配套的还有一个timeout事件，用来指定回调函数。

```
xhr.ontimeout = function(event){  
  
    alert('请求超时！');  
  
}
```

目前，Opera、Firefox和IE 10支持该属性，IE 8和IE 9的这个属性属于XDomainRequest对象，而Chrome和Safari还不支持。

#### 五、FormData对象

ajax操作往往用来传递表单数据。为了方便表单处理，HTML 5新增了一个FormData对象，可以模拟表单。

首先，新建一个FormData对象。

```
var formData = new FormData();
```

然后，为它添加表单项。

```
formData.append('username', '张三');  
  
formData.append('id', 123456);
```

最后，直接传送这个FormData对象。这与提交网页表单的效果，完全一样。

```
xhr.send(formData);
```

FormData对象也可以用来获取网页表单的值。

```
var form = document.getElementById('myform');  
  
var formData = new FormData(form);  
  
formData.append('secret', '123456'); // 添加一个表单项  
  
xhr.open('POST', form.action);  
  
xhr.send(formData);
```

## 六、上传文件

新版XMLHttpRequest对象，不仅可以发送文本信息，还可以上传文件。

假定files是一个"选择文件"的表单元素（input[type="file"]），我们将它装入FormData对象。

```
var formData = new FormData();  
  
for (var i = 0; i < files.length; i++) {  
  
    formData.append('files[]', files[i]);  
  
}
```

然后，发送这个FormData对象。

```
xhr.send(formData);
```

## 七、跨域资源共享 (CORS)

新版本的XMLHttpRequest对象，可以向不同域名的服务器发出HTTP请求。这叫做“[跨域资源共享](#)”（Cross-origin resource sharing，简称CORS）。

使用“跨域资源共享”的前提，是浏览器必须支持这个功能，而且服务器端必须同意这种“跨域”。如果能够满足上面的条件，则代码的写法与不跨域的请求完全一样。

```
xhr.open('GET', 'http://other.server/and/path/to/script');
```

目前，除了IE 8和IE 9，主流浏览器都支持CORS，IE 10也将支持这个功能。服务器端的设置，请参考[《Server-Side Access Control》](#)。

## 八、接收二进制数据（方法A：改写MimeType）

老版本的XMLHttpRequest对象，只能从服务器取回文本数据（否则它的名字就不用XML起首了），新版则可以取回二进制数据。

这里又分成两种做法。较老的做法是改写数据的MimeType，将服务器返回的二进制数据伪装成文本数据，并且告诉浏览器这是用户自定义的字符集。

```
xhr.overrideMimeType("text/plain; charset=x-user-defined");
```

然后，用responseText属性接收服务器返回的二进制数据。

```
var binStr = xhr.responseText;
```

由于这时，浏览器把它当做文本数据，所以还必须再一个个字节地还原成二进制数据。

```
for (var i = 0, len = binStr.length; i < len; ++i) {  
  
    var c = binStr.charCodeAt(i);  
  
    var byte = c & 0xff;  
  
}
```

最后一行的位运算“c & 0xff”，表示在每个字符的两个字节之中，只保留后一个字节，将前一个字节扔掉。原因是浏览器解读字符的时候，会把字符自动[解读](#)成Unicode的0xF700-0xF7ff区段。

## 八、接收二进制数据（方法B：responseType属性）

从服务器取回二进制数据，较新的方法是使用新增的responseType属性。如果服务器返回文本数据，这个属性的值是"TEXT"，这是默认值。较新的浏览器还支持其他值，也就是说，可以接收其他格式的数据。

你可以把responseType设为blob，表示服务器传回的是二进制对象。

```
var xhr = new XMLHttpRequest();

xhr.open('GET', '/path/to/image.png');

xhr.responseType = 'blob';
```

接收数据的时候，用浏览器自带的Blob对象即可。

```
var blob = new Blob([xhr.response], {type: 'image/png'});
```

注意，是读取xhr.response，而不是xhr.responseText。

你还可以将responseType设为arraybuffer，把二进制数据装在一个数组里。

```
var xhr = new XMLHttpRequest();

xhr.open('GET', '/path/to/image.png');

xhr.responseType = "arraybuffer";
```

接收数据的时候，需要遍历这个数组。

```
var arrayBuffer = xhr.response;

if (arrayBuffer) {

    var byteArray = new Uint8Array(arrayBuffer);

    for (var i = 0; i < byteArray.byteLength; i++) {

        // do something

    }

}
```

更详细的讨论，请看[Sending and Receiving Binary Data](#)。

## 九、进度信息

新版本的XMLHttpRequest对象，传送数据的时候，有一个progress事件，用来返回进度信息。

它分成上传和下载两种情况。下载的progress事件属于XMLHttpRequest对象，上传的progress事件属于XMLHttpRequest.upload对象。

我们先定义progress事件的回调函数。

```
xhr.onprogress = updateProgress;

xhr.upload.onprogress = updateProgress;
```

然后，在回调函数里面，使用这个事件的一些属性。

```
function updateProgress(event) {

    if (event.lengthComputable) {

        var percentComplete = event.loaded / event.total;

    }

}
```

上面的代码中，event.total是需要传输的总字节，event.loaded是已经传输的字节。如果event.lengthComputable不为真，则event.total等于0。

与progress事件相关的，还有其他五个事件，可以分别指定回调函数：

- \* load事件：传输成功完成。
- \* abort事件：传输被用户取消。
- \* error事件：传输中出现错误。
- \* loadstart事件：传输开始。
- \* loadend事件：传输结束，但是不知道成功还是失败。





## 十、阅读材料

1. [Introduction to XMLHttpRequest Level 2](#) : 新功能的综合介绍。
2. [New Tricks in XMLHttpRequest 2](#) : 一些用法的介绍。
3. [Using XMLHttpRequest](#) : 一些高级用法, 主要针对Firefox浏览器。
4. [HTTP Access Control](#) : CORS综述。
5. [DOM access control using cross-origin resource sharing](#) : CORS的9种HTTP头信息
6. [Server-Side Access Control](#) : 服务器端CORS设置。
7. [Enable CORS](#) : 服务端CORS设置。

( 完 )

### 文档信息

- 版权声明 : 自由转载-非商用-非衍生-保持署名 ( 创意共享3.0许可证 )
- 发表日期 : 2012年9月 8日
- 更多内容 : 档案 » JavaScript
- 文集 : 《前方的路》 , 《未来世界的幸存者》
- 社交媒体 :  twitter ,  weibo