MDN Web Docs
moz://a

Getting Started

这篇文章旨在帮助你了解AJAX基础，并提供两个可上手的简单案例供你学习。

什么是AJAX?

AJAX是异步的JavaScript和XML（**A**synchronous **J**avaScript **A**nd **X**ML）。简单点说，就是使用 `XMLHttpRequest` 对象与服务器通信。它可以使用JSON，XML，HTML和text文本等格式发送和接收数据。AJAX最吸引人的就是它的"异步"特性，也就是说它可以在不重新刷新页面的情况下与服务器通信，交换数据，或更新页面。

你可以使用AJAX最主要的两个特性做下列事：

- 在不重新加载页面的情况下发送请求给服务器。
- 接受并使用从服务器发来的数据。

Step 1 – 怎样发送http请求

为了使用JavaScript向服务器发送一个http请求，你需要一个包含必要函数功能的对象实例。这就是为什么会有 `XMLHttpRequest` 的原因。这是IE浏览器的ActiveX对象 `XMLHTTP` 的前身。 随后Mozilla，Safari和其他浏览器也都实现了类似的方法，被称为 `XMLHttpRequest` 。同时，微软也实现了XMLHttpRequest方法。

```
// Old compatibility code, no longer needed.
if (window.XMLHttpRequest) { // Mozilla, Safari, IE7+ ...
    httpRequest = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE 6 and older
    httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
}
```

> **Note: 上面代码只是简单版的如何创建一个XMLHttp实例。更实际的例子，请看本篇文章的step 3。**

发送一个请求后，你会收到响应。在这一阶段，你要告诉XMLHttp请求对象是由哪一个JavaScript函数处理响应，在设置了对象的 `onreadystatechange` 属性后给他命名，当请求状态改变时调用函数。

```
httpRequest.onreadystatechange = nameOfTheFunction;
```

要注意的是，函数名后没有参数，因为你把一个引用赋值给了函数，而不是真正的调用了它。 此外，如果不使用函数名的方式，你还可以用JavaScript的匿名函数响应处理的动作，就像下面这样：

```
httpRequest.onreadystatechange = function(){
  // Process the server response here.
};
```

接下来，声明当你接到响应后要做什么，你要发送一个实际的请求，通过调用HTTP请求对象的 `open()` 和 `send()` 方法，像下面这样：

```
httpRequest.open('GET', 'http://www.example.org/some.file', true);
httpRequest.send();
```


- `open()` 的第一个参数是HTTP请求方法 - 有GET，POST，HEAD以及服务器支持的其他方法。 保证这些方法一定要是大写字母，否则其他一些浏览器（比如FireFox）可能无法处理这个请求。更多关于HTTP的请求方法，可以查看 W3C specs。
- 第二个参数是你要发送的URL。由于安全原因，默认不能调用第三方URL域名。 确保你在页面中使用的是正确的域名，否则在调用 `open()` 方法是会有 "permission denied" 错误提示。一个容易犯的错误是你企图通过 `domain.tld` 访问网站， 而不是使用 `www.domain.tld`。如果你真的需要向另一个域名发送请求, 可以查看 HTTP access control。

- 第三个参数是可选的，用于设置请求是否是异步的。如果设为 `true` (默认值)，即开启异步，JavaScript就不会在此语句阻塞，使得用户能在服务器还没有响应的情况下与页面进行交互。

`send()` 方法的参数可以是任何你想发送给服务器的内容，如果是 `POST` 请求的话。发送表单数据时应该用服务器可以解析的格式，像查询语句：

```
"name=value&anothername="+encodeURIComponent(myVar)+"&so=on"
```

或者其他格式, 类似 `multipart/form-data`，JSON，XML等。

如果你使用 POST 数据，那就需要设置请求的MIME类型。比如，在调用 `send()` 方法获取表单数据前要有下面这个：

```
httpRequest.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
```

Step 2 – 处理服务器响应

在发送请求时，你提供的JavaScript函数名负责处理响应：

```
httpRequest.onreadystatechange = nameOfTheFunction;
```

这个函数应该做什么？首先，函数要检查请求的状态。如果状态的值是 `XMLHttpRequest.DONE` （对应的值是4），意味着服务器响应收到了并且是没问题的，然后就可以继续执行。

```
if (httpRequest.readyState === XMLHttpRequest.DONE) {
    // Everything is good, the response was received.
} else {
    // Not ready yet.
}
```

全部readyState状态值都在 XMLHTTPRequest.readyState，如下也是：

- 0 (未初始化) or (**请求还未初始化**)
- 1 (正在加载) or (**已建立服务器链接**)
- 2 (加载成功) or (**请求已接受**)
- 3 (交互) or (**正在处理请求**)
- 4 (完成) or (**请求已完成并且响应已准备好**)

(Source)

接下来，点击HTTP响应的 response code。 可能的响应码都已经列在 W3C这个列表里。在下面的例子中，我们通过检查响应码 `200 OK` 判断AJAX有没有成功。

```
if (httpRequest.status === 200) {
    // Perfect!
} else {
    // There was a problem with the request.
    // For example, the response may have a 404 (Not Found)
    // or 500 (Internal Server Error) response code.
}
```


在检查完请求状态和HTTP响应码后， 你就可以用服务器返回的数据做任何你想做的了。你有两个方法去访问这些数据：

- `httpRequest.responseText` – 服务器以文本字符的形式返回
- `httpRequest.responseXML` – 以 XMLDocument 对象方式返回，之后就可以使用JavaScript来处理

注意上面这一步只在你发起异步请求时有效（即 `open()` 的第三个参数未特别指定或设为 `true`）。如果你发起的是**同步**请求则不必使用函数，但是非常不推荐这样子做，它的用户体验很糟糕。

Step 3 – 一个简单的例子

让我们把所有的知识都集中起来做一个简单的HTTP请求。这个JavaScript会请求一个HTML文档 `test.html`，包含 "I'm a test" 内容。然后我们 `alert()` 响应的内容。注意这个例子我们只是用了JavaScript，没有用jQuery。而且，HTML，XML和PHP文件都要放在用一个目录下。

```
<button id="ajaxButton" type="button">Make a request</button>

<script>
(function() {
  var httpRequest;
  document.getElementById("ajaxButton").addEventListener('click', makeRequest);

  function makeRequest() {
    httpRequest = new XMLHttpRequest();

    if (!httpRequest) {
      alert('Giving up :( Cannot create an XMLHTTP instance');
      return false;
    }
    httpRequest.onreadystatechange = alertContents;
    httpRequest.open('GET', 'test.html');
    httpRequest.send();
  }

  function alertContents() {
    if (httpRequest.readyState === XMLHttpRequest.DONE) {
      if (httpRequest.status === 200) {
        alert(httpRequest.responseText);
      } else {
        alert('There was a problem with the request.');
      }
    }
  }
})();
</script>
```

在这个例子中：

- 用户点击 "Make a request" 按钮；
- 事件处理调用 `makeRequest()` 函数；
- 请求已通过然后（`onreadystatechange`）传给 `alertContents()` 执行。
- `alertContents()` 检查返回的响应是否OK，然后 `alert()` `test.html` 文件内容。

> **Note**: 如果你向一个代码片段发送请求，将返回XML，而不是静态XML文件，在IE浏览器上则必须要设置响应头才能正常工作。如果不设置响应头为 `Content-Type:application/xml` ，IE浏览器会在你访问XML元素时抛出 "Object Expected" 错误。

> **Note 2**: 如果不设置响应头 `Cache-Control: no-cache` 浏览器将会把响应缓存下来而且再也无法重新提交请求。你也可以添加一个总是不同的 GET 参数，比如时间戳或者随机数 (详情见 bypassing the cache)

> **Note 3**: 如果变量 `httpRequest` 在全局范围内使用，它会在 `makeRequest()` 函数中被相互覆盖，从而导致资源竞争。为了避免这个情况，请在包含 AJAX 函数的闭包中声明 `httpRequest` 变量。

在通信错误的事件中（例如服务器宕机），在访问响应状态 onreadystatechange 方法中会抛出一个例外。
为了缓和这种情况，则可以使用 `try...catch` 把 `if...then` 语句包裹起来。

```
function alertContents() {
  try {
    if (httpRequest.readyState === XMLHttpRequest.DONE) {
      if (httpRequest.status === 200) {
        alert(httpRequest.responseText);
      } else {
        alert('There was a problem with the request.');
      }
    }
  }
  catch( e ) {
    alert('Caught Exception: ' + e.description);
  }
}
```

Step 4 – 处理 XML 响应

在上一个例子中，在收到HTTP请求的响应后我们会请求对象的 `responseText` 属性，包含 `test.html` 文件的内容。现在我们试试 `responseXML` 属性。

首先，我们创建一个稍后将要请求的有效的 XML 文档。文档（`test.html`）包含以下内容：

```
<?xml version="1.0" ?>
<root>
    I'm a test.
</root>
```

在脚本里我们只需要把请求行改为：

```
...
onclick="makeRequest('test.xml')">
...
```

然后在 `alertContents()` 里，我们把 `alert(httpRequest.responseText)` 改为：

```
var xmldoc = httpRequest.responseXML;
var root_node = xmldoc.getElementsByTagName('root').item(0);
alert(root_node.firstChild.data);
```

这部分代码采用 `responseXML` 提供的 `XMLDocument` 对象，并使用 DOM 方法访问 XML 文档中包含的一些数据。你可以在 这里查看 `test.xml` 并且在 这里更新测试代码。

Step 5 – 处理数据

最后，我们发送一个数据给服务器并收到响应。这次我们用 JavaScript 请求动态页面， `test.php` 并返回一个计算后的字符串 - "Hello, [user date]"，并用 `alert()` 出来。

首先要添加一个文本到 HTML 中以方便用户输入名字：

```
<label>Your name:
  <input type="text" id="ajaxTextbox" />
</label>
<span id="ajaxButton" style="cursor: pointer; text-decoration: underline">
  Make a request
</span>
```


还要添加事件处理程序，从表单中获取用户数据连同服务器端的UTL一并发送给 makeRequest() 函数：

```
 document.getElementById("ajaxButton").onclick = function() {
     var userName = document.getElementById("ajaxTextbox").value;
     makeRequest('test.php',userName);
 };
```

我们还要修改 `makeRequest()` 让它接受用户数据并将其发给服务器。把请求方法从 `GET` 改为 `POST`，把数据作为参数让 `httpRequest.send()` 调用。

```
 function makeRequest(url, userName) {

   ...

  httpRequest.onreadystatechange = alertContents;
  httpRequest.open('POST', url);
  httpRequest.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
  httpRequest.send('userName=' + encodeURIComponent(userName));
 }
```

如果这就是服务器返回的全部内容， `alertContents()` 函数可以使用 step 3 中的相同函数写。可是，服务器会返回计算后的内容和原内容。所以，如果用户输入 "Jane" 在输入框中，那服务器就会返回如下内容：

```
{"userData":"Jane","computedString":"Hi, Jane!"}
```

为了在 `alertContents()` 中使用这个数据，我们可不能只是alert `responseText` ，我们要解析它并 alert `conputedString`，我们想要的属性：

```
function alertContents() {
  if (httpRequest.readyState === XMLHttpRequest.DONE) {
    if (httpRequest.status === 200) {
      var response = JSON.parse(httpRequest.responseText);
      alert(response.computedString);
    } else {
      alert('There was a problem with the request.');
    }
  }
}
```

`test.php` 文件应该包含以下内容：

```
$name = (isset($_POST['userName'])) ? $_POST['userName'] : 'no name';
$computedString = "Hi, " . $name;
$array = ['userName' => $name, 'computedString' => $computedString];
echo json_encode($array);
```

想获取更多 DOM 方法，可以查看 `Mozilla's DOM implementation` 文档.

Last modified: 2020年5月9日, by MDN contributors