AJAX là gì?

AJAX là viết tắt của JavaScript và XML không đồng bộ (Asynchronous JavaScript and XML).  
AJAX là một kỹ thuật mới để tạo ra các ứng dụng web tốt hơn, nhanh hơn và tương tác hơn  
với sự trợ giúp của XML, HTML, CSS và Java Script.

* Ajax sử dụng XHTML cho nội dung, CSS cho trình bày, cùng với Document Object Model vàJavaScript để hiển thị nội dung động.
* Các ứng dụng web thông thường truyền thông tin đến và server gửi các yêu cầu để đồng bộ.Điều này có nghĩa là khi bạn điền vào một biểu mẫu, nhấn gửi và được chuyển đến mộtrang mới có thông tin mới từ server.
* Với AJAX, khi bạn nhấn gửi, JavaScript sẽ gửi yêu cầu tới server, hiển thị kết quả và cập nhậttại màn hình hiện tại. Theo nghĩa thuần túy nhất, người dùng sẽ không bao giờ biết rằng bất cứ điều gì được truyền tới server.
* XML thường được sử dụng làm định dạng để nhận dữ liệu server, mặc dù bất kỳ định dạng nào, kể cả văn bản thuần túy đều có thể được sử dụng.
* AJAX là một công nghệ trình duyệt web độc lập với phần mềm web server.
* Người dùng có thể tiếp tục sử dụng ứng dụng trong khi chương trình client tiếp tục xử lý yêu cầu thông tin từ server.
* Tương tác trực quan và tự nhiên của người dùng. Không yêu cầu nhấp chuột, chuyển động của chuột như là một trình kích hoạt sự kiện.
* Điều khiển dữ liệu có thể trái ngược với hướng trang

1. Ajax-Action

Một sự kiện client xảy ra.  
Một đối tượng XMLHttpRequest được tạo ra.  
Đối tượng XMLHttpRequest được cấu hình.  
Đối tượng XMLHttpRequest thực hiện một yêu cầu không đồng bộ tới Webserver.  
Webserver trả về kết quả chứa tài liệu XML.  
Đối tượng XMLHttpRequest gọi hàm callback () và xử lý kết quả.  
HTML DOM được cập nhật.  
Sau đây sẽ cụ thể từng bước một:

**Một sự kiện client xảy ra**

Một hàm JavaScript được gọi là kết quả của một sự kiện.  
Ví dụ - hàm JavaScript validateUserId () được ánh xạ như một trình xử lý sự kiện cho một sự  
kiện onkeyup trên trường biểu mẫu đầu vào có id = "userid"

<input type = "text" size = "20" id = "userid" name = "id" onkeyup = "validateUserId();">

**Đối tượng XMLHttpRequest được tạo**

var ajaxRequest; *// The variable that makes Ajax possible!*

function ajaxFunction() {  
 try {  
// Opera 8.0+, Firefox, Safari  
 ajaxRequest = new XMLHttpRequest();  
 } catch (e) {  
// Internet Explorer Browsers  
 try {  
 ajaxRequest = new ActiveXObject("Msxml2.XMLHTTP");  
 } catch (e) {  
 try {  
 ajaxRequest = new ActiveXObject("Microsoft.XMLHTTP");  
 } catch (e) {  
// Something went wrong  
 alert("Your browser broke!");  
 return false;  
 }  
 }  
 }  
}

**Đối tượng XMLHttpRequest được cấu hình**

Trong bước này, ta sẽ viết một hàm sẽ được kích hoạt bởi sự kiện client và hàm callRequest () sẽ được đăng ký.

function validateUserId() {

ajaxFunction();  
// Here processRequest() is the callback function.  
 ajaxRequest.onreadystatechange = processRequest;  
 if (!target) target = document.getElementById("userid");  
 var url = "validate?id=" + escape(target.value);  
 ajaxRequest.open("GET", url, true);  
 ajaxRequest.send(null);  
}

**Thực hiện yêu cầu không đồng bộ cho  sever web**

Theo mã nguồn ở trên, bằng cách sử dụng đối tượng XMLHttpRequest ajaxRequest , client sẽ gửi một request tới Server

Giả sử bạn nhập Zara vào hộp userid, sau đó trong yêu cầu ở trên, URL được đặt thành "validate? Id = Zara"

**Server Web Trả về kết quả chứa tài liệu XML**

Bạn có thể triển khai kịch bản phía server của mình bằng bất kỳ ngôn ngữ nào, tuy nhiên, logic của nó phải như sau.

• Nhận yêu cầu từ khách hàng.

• Phân tích cú pháp đầu vào từ máy khách.

• Thực hiện xử lý.

• Gửi đầu ra cho máy khách.

Ta giả định rằng bạn sẽ viết một đoạn mã servlet như sau.

public void doGet(HttpServletRequest request,HttpServletResponse response) throws IOException, ServletException {

String targetId = request.getParameter("id");  
  
 if ((targetId != null) && !accounts.containsKey(targetId.trim())) {  
 response.setContentType("text/xml");  
 response.setHeader("Cache-Control", "no-cache");  
 response.getWriter().write("<valid>true</valid>");  
 } else {  
 response.setContentType("text/xml");  
 response.setHeader("Cache-Control", "no-cache");  
 response.getWriter().write("<valid>false</valid>");  
 }  
}

**Hàm gọi lại processRequest () được gọi**

Đối tượng XMLHttpRequest được cấu hình để gọi hàm processRequest () khi có sự thay đổi trạng thái đối với readyState của đối tượng XMLHttpRequest. Bây giờ chức năng này sẽ nhận được kết quả từ máy chủ và sẽ thực hiện xử lý yêu cầu. Như trong ví dụ sau, nó đặt một thông báo biến trên true hoặc false dựa trên giá trị trả về từ máy chủ Web.

function processRequest() {  
if (req.readyState == 4) {  
 if (req.status == 200) {  
 var message = ...;  
 ...  
}

**HTML DOM được cập nhật**

Đây là bước cuối cùng và trong bước này, trang HTML của bạn sẽ được cập nhật.

JavaScript nhận tham chiếu đến bất kỳ phần tử nào trong một trang bằng cách gọi sử dụng API DOM

 document.getElementById("userIdMessage"),  
// where "userIdMessage" is the ID attribute  
// of an element appearing in the HTML document

JavaScript bây giờ có thể được sử dụng để sửa đổi các thuộc tính của phần tử; sửa đổi các thuộc tính của phần tử; hoặc thêm, xóa hoặc sửa đổi các phần tử con. Đây là một ví dụ:

<script type = "text/javascript">  
 <!--  
 function setMessageUsingDOM(message) {  
 var userMessageElement = document.getElementById("userIdMessage");  
 var messageText;  
 if (message == "false") {  
 userMessageElement.style.color = "red";  
 messageText = "Invalid User Id";  
 } else {  
 userMessageElement.style.color = "green";  
 messageText = "Valid User Id";  
 }  
 var messageBody = document.createTextNode(messageText);  
*// if the messageBody element has been created simple*  
*// replace it otherwise append the new element*  
 *if (userMessageElement.childNodes[0]) {*  
 *userMessageElement.replaceChild(messageBody, userMessageElement.childNodes[0]);*  
 *} else {*  
 *userMessageElement.appendChild(messageBody);*  
 *}*  
 *}*  
*-->*  
</script>  
<body>  
<div id = "userIdMessage"><div>  
</body>

2. AJAX - XMLHttpRequest

Đối tượng XMLHttpRequest là chìa khóa của AJAX. Nó đã có sẵn kể từ khi Internet Explorer 5.5 được phát hành vào tháng 7 năm 2000, nhưng không được phát hiện đầy đủ cho đến khi AJAX và Web 2.0 năm 2005 trở nên phổ biến.

XMLHttpRequest (XHR) là một API có thể được sử dụng bởi JavaScript, JScript, VBScript và các ngôn ngữ kịch bản trình duyệt web khác để truyền và thao tác dữ liệu XML đến và từ máy chủ web bằng [HTTP](https://james.codegym.vn/mod/page/view.php?id=1716), thiết lập kênh kết nối độc lập giữa Client-Side của trang web và Phía máy chủ.

Dữ liệu được trả về từ các cuộc gọi XMLHttpRequest thường sẽ được cung cấp bởi các cơ sở dữ liệu phía sau. Bên cạnh XML, XMLHttpRequest có thể được sử dụng để tìm nạp dữ liệu ở các định dạng khác, ví dụ: JSON hoặc thậm chí là văn bản thuần túy.

Bạn đã thấy một vài ví dụ về cách tạo một đối tượng XMLHttpRequest.

Dưới đây là một số phương pháp và thuộc tính mà bạn phải làm quen.

**Các phương thức XMLHttpRequest**

* abort() Hủy yêu cầu hiện tại.
* getAllResponseHeaders() Trả về tập hợp đầy đủ các tiêu đề [HTTP](https://james.codegym.vn/mod/page/view.php?id=1716) dưới dạng một chuỗi.
* getResponseHeader( headerName ) Trả về giá trị của tiêu đề [HTTP](https://james.codegym.vn/mod/page/view.php?id=1716) được chỉ định.
* open( method, URL )
* open( method, URL, async )
* open( method, URL, async, userName )
* open( method, URL, async, userName, password )

Chỉ định phương thức, URL và các thuộc tính tùy chọn khác của một request.

Tham số phương thức có thể có giá trị "GET", "POST" hoặc "HEAD", hoặc có thể là các phương thức [HTTP](https://james.codegym.vn/mod/page/view.php?id=1716) khác như "PUT" và "DELETE" (chủ yếu được sử dụng trong các ứng dụng REST).

Tham số "async" chỉ định liệu yêu cầu có nên được xử lý không đồng bộ hay không. "true" có nghĩa là quá trình xử lý script tiếp tục sau phương thức send () mà không phải chờ phản hồi và "false" có nghĩa là tập lệnh chờ phản hồi trước khi tiếp tục xử lý tập lệnh.

* send( content ) Gửi yêu cầu.
* setRequestHeader( label, value ) Thêm 1 cặp label/value vào [HTTP](https://james.codegym.vn/mod/page/view.php?id=1716) header để gửi.

**Các thuộc tính XMLHttpRequest**

* onreadystatechange

Một Event Handler lắng nghe và xử lý một sự kiện khi có thay đổi về trạng thái diễn ra.

* readyState

Thuộc tính này định nghĩa trạng thái hiện tại của đối tượng XMLHttpRequest.

Bảng dưới đây liệt kê các giá trị cho thuộc tính readyState

|  |  |
| --- | --- |
| Trạng thái | Miêu tả |
| 0 | Requet chưa được khởi tạo. |
| 1 | Request đã được thiết lập. |
| 2 | Request đã được gửi. |
| 3 | Request đang được xử lý. |
| 4 | Kết thúc Request. |

readyState = 0 Sau khi bạn đã tạo đối tượng XMLHttpRequest, nhưng trước khi bạn đã gọi phương thức open().

readyState = 1 Sau khi bạn đã gọi phương thức open(), nhưng trước khi bạn đã gọi send ().

readyState = 2 Sau khi bạn đã gọi send().

readyState = 3 Sau khi trình duyệt đã thiết lập một giao tiếp với máy chủ, nhưng trước khi máy chủ hoàn tất phản hồi.

readyState = 4 Sau khi yêu cầu đã được hoàn thành và dữ liệu phản hồi đã được nhận hoàn toàn từ máy chủ.

* responseText

Trả về phản hồi dưới dạng chuỗi.

* responseXML

Trả về phản hồi dưới dạng XML. Thuộc tính này trả về một đối tượng tài liệu XML, có thể được kiểm tra và phân tích cú pháp bằng cách sử dụng các phương thức và các thuộc tính cây nút của W3C DOM.

* status

Trả về trạng thái dưới dạng số (e.g., 404 for "Not Found" and 200 for "OK").

* statusText

Trả về trạng thái dưới dạng chuỗi (e.g., "Not Found" or "OK").

Last modified: Monday, 2 August 2021, 11:30 AM