What is AJAX?

AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.

AJAX is not a programming language.

AJAX just uses a combination of:

* A browser built-in XMLHttpRequest object (to request data from a web server)
* JavaScript and HTML DOM (to display or use the data)

AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

How AJAX Works



* 1. An event occurs in a web page (the page is loaded, a button is clicked)
* 2. An XMLHttpRequest object is created by JavaScript
* 3. The XMLHttpRequest object sends a request to a web server
* 4. The server processes the request
* 5. The server sends a response back to the web page
* 6. The response is read by JavaScript
* 7. Proper action (like page update) is performed by JavaScript

Example1:

<!DOCTYPE html>

<html>

<body>

<div id="demo">

<h2>The XMLHttpRequest Object</h2>

<button type="button" onclick="loadDoc()">Change Content</button>

</div>

<script>

function loadDoc() {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("demo").innerHTML =

this.responseText;

}

};

xhttp.open("GET", "ajax\_info.txt", true);

xhttp.send();

}

</script>

</body>

</html>

# **AJAX - The XMLHttpRequest Object**

The keystone of AJAX is the XMLHttpRequest object.

## The XMLHttpRequest Object

All modern browsers support the XMLHttpRequest object.

The XMLHttpRequest object can be used to exchange data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

## Create an XMLHttpRequest Object

All modern browsers (Chrome, Firefox, IE7+, Edge, Safari, Opera) have a built-in XMLHttpRequest object.

Syntax for creating an XMLHttpRequest object:

*variable*= new XMLHttpRequest();

### **Example**

var xhttp = new XMLHttpRequest();

## Access Across Domains

For security reasons, modern browsers do not allow access across domains.

This means that both the web page and the XML file it tries to load, must be located on the same server.

The examples on W3Schools all open XML files located on the W3Schools domain.

If you want to use the example above on one of your own web pages, the XML files you load must be located on your own server.

## Older Browsers (IE5 and IE6)

Old versions of Internet Explorer (5/6) use an ActiveX object instead of the XMLHttpRequest object:

*variable*= new ActiveXObject("Microsoft.XMLHTTP");

To handle IE5 and IE6, check if the browser supports the XMLHttpRequest object, or else create an ActiveX object:

### **Example**

<!DOCTYPE html>

<html>

<body>

<h2>The XMLHttpRequest Object</h2>

<p id="demo">Let AJAX change this text.</p>

<button type="button" onclick="loadDoc()">Change Content</button>

<script>

function loadDoc() {

var xhttp;

if (window.XMLHttpRequest) {

// code for modern browsers

xhttp = new XMLHttpRequest();

} else {

// code for IE6, IE5

xhttp = new ActiveXObject("Microsoft.XMLHTTP");

}

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("demo").innerHTML = this.responseText;

}

};

xhttp.open("GET", "ajax\_info.txt", true);

xhttp.send();

}

</script>

</body>

</html>

## XMLHttpRequest Object Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| new XMLHttpRequest() | Creates a new XMLHttpRequest object |
| abort() | Cancels the current request |
| getAllResponseHeaders() | Returns header information |
| getResponseHeader() | Returns specific header information |
| open(method, url, async, user, psw) | Specifies the request  method: the request type GET or POST url: the file location async: true (asynchronous) or false (synchronous) user: optional user name psw: optional password |
| send() | Sends the request to the server Used for GET requests |
| send(string) | Sends the request to the server. Used for POST requests |
| setRequestHeader() | Adds a label/value pair to the header to be sent |

## XMLHttpRequest Object Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| Onreadystatechange | Defines a function to be called when the readyState  property changes |
| readyState | Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready |
| responseText | Returns the response data as a string |
| responseXML | Returns the response data as XML data |
| Status | Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the [Http Messages Reference](https://www.w3schools.com/tags/ref_httpmessages.asp) |
| statusText | Returns the status-text (e.g. "OK" or "Not Found") |

# **AJAX - Send a Request To a Server**

The XMLHttpRequest object is used to exchange data with a server.

Send a Request To a Server

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

xhttp.open("GET", "ajax\_info.txt", true);  
xhttp.send();

|  |  |
| --- | --- |
| **Method** | **Description** |
| open(*method, url, async*) | Specifies the type of request  *method*: the type of request: GET or POST *url*: the server (file) location *async*: true (asynchronous) or false (synchronous) |
| send() | Sends the request to the server (used for GET) |
| send(*string*) | Sends the request to the server (used for POST) |

GET or POST?

GET is simpler and faster than POST, and can be used in most cases.

However, always use POST requests when:

* A cached file is not an option (update a file or database on the server).
* Sending a large amount of data to the server (POST has no size limitations).
* Sending user input (which can contain unknown characters), POST is more robust and secure than GET.

GET Requests

A simple GET request:

<!DOCTYPE html>

<html>

<body>

<h2>The XMLHttpRequest Object</h2>

<button type="button" onclick="loadDoc()">Request data</button>

<p id="demo"></p>

<script>

function loadDoc() {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("demo").innerHTML = this.responseText;

}

};

xhttp.open("GET", "demo\_get.asp", true);

xhttp.send();

}

</script>

</body>

</html>

Example:

If you want to send information with the GET method, add the information to the URL:

<!DOCTYPE html>

<html>

<body>

<h2>The XMLHttpRequest Object</h2>

<button type="button" onclick="loadDoc()">Request data</button>

<p id="demo"></p>

<script>

function loadDoc() {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("demo").innerHTML = this.responseText;

}

};

xhttp.open("GET", "demo\_get2.asp?fname=Henry&lname=Ford", true);

xhttp.send();

}

</script>

</body>

</html>

## POST Requests

A simple POST request:

<!DOCTYPE html>

<html>

<body>

<h2>The XMLHttpRequest Object</h2>

<button type="button" onclick="loadDoc()">Request data</button>

<p id="demo"></p>

<script>

function loadDoc() {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("demo").innerHTML = this.responseText;

}

};

xhttp.open("POST", "demo\_post2.asp", true);

xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

xhttp.send("fname=Henry&lname=Ford");

}

</script>

</body>

</html>

|  |  |
| --- | --- |
| **Method** | **Description** |
| setRequestHeader(*header, value*) | Adds HTTP headers to the request  *header*: specifies the header name *value*: specifies the header value |

## The url - A File On a Server

The url parameter of the open() method, is an address to a file on a server:

xhttp.open("GET", "ajax\_test.asp", true);

The file can be any kind of file, like .txt and .xml, or server scripting files like .asp and .php (which can perform actions on the server before sending the response back).

## Asynchronous - True or False?

Server requests should be sent asynchronously.

The async parameter of the open() method should be set to true:

xhttp.open("GET", "ajax\_test.asp", true);

By sending asynchronously, the JavaScript does not have to wait for the server response, but can instead:

* execute other scripts while waiting for server response
* deal with the response after the response is ready

## The onreadystatechange Property

With the XMLHttpRequest object you can define a function to be executed when the request receives an answer.

The function is defined in the onreadystatechange property of the XMLHttpResponse object:

### **Example**

xhttp.onreadystatechange = function() {  
  if (this.readyState == 4 && this.status == 200) {  
    document.getElementById("demo").innerHTML = this.responseText;  
  }  
};  
xhttp.open("GET", "ajax\_info.txt", true);  
xhttp.send();

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_ajax_first)

You will learn more about onreadystatechange in a later chapter.

## Synchronous Request

To execute a synchronous request, change the third parameter in the open() method to false:

xhttp.open("GET", "ajax\_info.txt", false);

Sometimes async = false are used for quick testing. You will also find synchronous requests in older JavaScript code.

Since the code will wait for server completion, there is no need for an onreadystatechange function:

### **Example**

xhttp.open("GET", "ajax\_info.txt", false);  
xhttp.send();  
document.getElementById("demo").innerHTML = xhttp.responseText;

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_ajax_asyncfalse)

Synchronous XMLHttpRequest (async = false) is not recommended because the JavaScript will stop executing until the server response is ready. If the server is busy or slow, the application will hang or stop.

Synchronous XMLHttpRequest is in the process of being removed from the web standard, but this process can take many years.

Modern developer tools are encouraged to warn about using synchronous requests and may throw an InvalidAccessError exception when it occurs.

# **AJAX XML Example**

AJAX can be used for interactive communication with an XML file.

## AJAX XML Example

The following example will demonstrate how a web page can fetch information from an XML file with AJAX:

<!DOCTYPE html>

<html>

<style>

table,th,td {

border : 1px solid black;

border-collapse: collapse;

}

th,td {

padding: 5px;

}

</style>

<body>

<h2>The XMLHttpRequest Object</h2>

<button type="button" onclick="loadDoc()">Get my CD collection</button>

<br><br>

<table id="demo"></table>

<script>

function loadDoc() {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

myFunction(this);

}

};

xhttp.open("GET", "cd\_catalog.xml", true);

xhttp.send();

}

function myFunction(xml) {

var i;

var xmlDoc = xml.responseXML;

var table="<tr><th>Artist</th><th>Title</th></tr>";

var x = xmlDoc.getElementsByTagName("CD");

for (i = 0; i <x.length; i++) {

table += "<tr><td>" +

x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +

"</td><td>" +

x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +

"</td></tr>";

}

document.getElementById("demo").innerHTML = table;

}

</script>

</body>

</html>

## Example Explained

When a user clicks on the "Get CD info" button above, the loadDoc() function is executed.

The loadDoc() function creates an XMLHttpRequest object, adds the function to be executed when the server response is ready, and sends the request off to the server.

When the server response is ready, an HTML table is built, nodes (elements) are extracted from the XML file, and it finally updates the element "demo" with the HTML table filled with XML data:

function loadDoc() {  
  var xhttp = new XMLHttpRequest();  
  xhttp.onreadystatechange = function() {  
    if (this.readyState == 4 && this.status == 200) {  
    myFunction(this);  
    }  
  };  
  xhttp.open("GET", "cd\_catalog.xml", true);  
  xhttp.send();  
}  
function myFunction(xml) {  
  var i;  
  var xmlDoc = xml.responseXML;  
  var table="<tr><th>Artist</th><th>Title</th></tr>";  
  var x = xmlDoc.getElementsByTagName("CD");  
  for (i = 0; i <x.length; i++) {  
    table += "<tr><td>" +  
    x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +  
    "</td><td>" +  
    x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +  
    "</td></tr>";  
  }  
  document.getElementById("demo").innerHTML = table;  
}

## The XML File

The XML file used in the example above looks like this: "[cd\_catalog.xml](https://www.w3schools.com/js/cd_catalog.xml)".

This XML file does not appear to have any style information associated with it. The document tree is shown below.

<CATALOG>

<CD>

<TITLE>Empire Burlesque</TITLE>

<ARTIST>Bob Dylan</ARTIST>

<COUNTRY>USA</COUNTRY>

<COMPANY>Columbia</COMPANY>

<PRICE>10.90</PRICE>

<YEAR>1985</YEAR>

</CD>

<CD>

<TITLE>Hide your heart</TITLE>

<ARTIST>Bonnie Tyler</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>CBS Records</COMPANY>

<PRICE>9.90</PRICE>

<YEAR>1988</YEAR>

</CD>

<CD>

<TITLE>Greatest Hits</TITLE>

<ARTIST>Dolly Parton</ARTIST>

<COUNTRY>USA</COUNTRY>

<COMPANY>RCA</COMPANY>

<PRICE>9.90</PRICE>

<YEAR>1982</YEAR>

</CD>

<CD>

<TITLE>Still got the blues</TITLE>

<ARTIST>Gary Moore</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>Virgin records</COMPANY>

<PRICE>10.20</PRICE>

<YEAR>1990</YEAR>

</CD>

<CD>

<TITLE>Eros</TITLE>

<ARTIST>Eros Ramazzotti</ARTIST>

<COUNTRY>EU</COUNTRY>

<COMPANY>BMG</COMPANY>

<PRICE>9.90</PRICE>

<YEAR>1997</YEAR>

</CD>

<CD>

<TITLE>One night only</TITLE>

<ARTIST>Bee Gees</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>Polydor</COMPANY>

<PRICE>10.90</PRICE>

<YEAR>1998</YEAR>

</CD>

<CD>

<TITLE>Sylvias Mother</TITLE>

<ARTIST>Dr.Hook</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>CBS</COMPANY>

<PRICE>8.10</PRICE>

<YEAR>1973</YEAR>

</CD>

<CD>

<TITLE>Maggie May</TITLE>

<ARTIST>Rod Stewart</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>Pickwick</COMPANY>

<PRICE>8.50</PRICE>

<YEAR>1990</YEAR>

</CD>

<CD>

<TITLE>Romanza</TITLE>

<ARTIST>Andrea Bocelli</ARTIST>

<COUNTRY>EU</COUNTRY>

<COMPANY>Polydor</COMPANY>

<PRICE>10.80</PRICE>

<YEAR>1996</YEAR>

</CD>

<CD>

<TITLE>When a man loves a woman</TITLE>

<ARTIST>Percy Sledge</ARTIST>

<COUNTRY>USA</COUNTRY>

<COMPANY>Atlantic</COMPANY>

<PRICE>8.70</PRICE>

<YEAR>1987</YEAR>

</CD>

<CD>

<TITLE>Black angel</TITLE>

<ARTIST>Savage Rose</ARTIST>

<COUNTRY>EU</COUNTRY>

<COMPANY>Mega</COMPANY>

<PRICE>10.90</PRICE>

<YEAR>1995</YEAR>

</CD>

<CD>

<TITLE>1999 Grammy Nominees</TITLE>

<ARTIST>Many</ARTIST>

<COUNTRY>USA</COUNTRY>

<COMPANY>Grammy</COMPANY>

<PRICE>10.20</PRICE>

<YEAR>1999</YEAR>

</CD>

<CD>

<TITLE>For the good times</TITLE>

<ARTIST>Kenny Rogers</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>Mucik Master</COMPANY>

<PRICE>8.70</PRICE>

<YEAR>1995</YEAR>

</CD>

<CD>

<TITLE>Big Willie style</TITLE>

<ARTIST>Will Smith</ARTIST>

<COUNTRY>USA</COUNTRY>

<COMPANY>Columbia</COMPANY>

<PRICE>9.90</PRICE>

<YEAR>1997</YEAR>

</CD>

<CD>

<TITLE>Tupelo Honey</TITLE>

<ARTIST>Van Morrison</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>Polydor</COMPANY>

<PRICE>8.20</PRICE>

<YEAR>1971</YEAR>

</CD>

<CD>

<TITLE>Soulsville</TITLE>

<ARTIST>Jorn Hoel</ARTIST>

<COUNTRY>Norway</COUNTRY>

<COMPANY>WEA</COMPANY>

<PRICE>7.90</PRICE>

<YEAR>1996</YEAR>

</CD>

<CD>

<TITLE>The very best of</TITLE>

<ARTIST>Cat Stevens</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>Island</COMPANY>

<PRICE>8.90</PRICE>

<YEAR>1990</YEAR>

</CD>

<CD>

<TITLE>Stop</TITLE>

<ARTIST>Sam Brown</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>A and M</COMPANY>

<PRICE>8.90</PRICE>

<YEAR>1988</YEAR>

</CD>

<CD>

<TITLE>Bridge of Spies</TITLE>

<ARTIST>T'Pau</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>Siren</COMPANY>

<PRICE>7.90</PRICE>

<YEAR>1987</YEAR>

</CD>

<CD>

<TITLE>Private Dancer</TITLE>

<ARTIST>Tina Turner</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>Capitol</COMPANY>

<PRICE>8.90</PRICE>

<YEAR>1983</YEAR>

</CD>

<CD>

<TITLE>Midt om natten</TITLE>

<ARTIST>Kim Larsen</ARTIST>

<COUNTRY>EU</COUNTRY>

<COMPANY>Medley</COMPANY>

<PRICE>7.80</PRICE>

<YEAR>1983</YEAR>

</CD>

<CD>

<TITLE>Pavarotti Gala Concert</TITLE>

<ARTIST>Luciano Pavarotti</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>DECCA</COMPANY>

<PRICE>9.90</PRICE>

<YEAR>1991</YEAR>

</CD>

<CD>

<TITLE>The dock of the bay</TITLE>

<ARTIST>Otis Redding</ARTIST>

<COUNTRY>USA</COUNTRY>

<COMPANY>Stax Records</COMPANY>

<PRICE>7.90</PRICE>

<YEAR>1968</YEAR>

</CD>

<CD>

<TITLE>Picture book</TITLE>

<ARTIST>Simply Red</ARTIST>

<COUNTRY>EU</COUNTRY>

<COMPANY>Elektra</COMPANY>

<PRICE>7.20</PRICE>

<YEAR>1985</YEAR>

</CD>

<CD>

<TITLE>Red</TITLE>

<ARTIST>The Communards</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>London</COMPANY>

<PRICE>7.80</PRICE>

<YEAR>1987</YEAR>

</CD>

<CD>

<TITLE>Unchain my heart</TITLE>

<ARTIST>Joe Cocker</ARTIST>

<COUNTRY>USA</COUNTRY>

<COMPANY>EMI</COMPANY>

<PRICE>8.20</PRICE>

<YEAR>1987</YEAR>

</CD>

</CATALOG>

# **AJAX PHP Example**

AJAX is used to create more interactive applications.

## AJAX PHP Example

The following example demonstrates how a web page can communicate with a web server while a user types characters in an input field:

### **Example**

<!DOCTYPE html>

<html>

<body>

<h2>The XMLHttpRequest Object</h2>

<h3>Start typing a name in the input field below:</h3>

<p>Suggestions: <span id="txtHint"></span></p>

<p>First name: <input type="text" id="txt1" onkeyup="showHint(this.value)"></p>

<script>

function showHint(str) {

var xhttp;

if (str.length == 0) {

document.getElementById("txtHint").innerHTML = "";

return;

}

xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("txtHint").innerHTML = this.responseText;

}

};

xhttp.open("GET", "gethint.php?q="+str, true);

xhttp.send();

}

</script>

</body>

</html>

Code explanation:

First, check if the input field is empty (str.length == 0). If it is, clear the content of the txtHint placeholder and exit the function.

However, if the input field is not empty, do the following:

* Create an XMLHttpRequest object
* Create the function to be executed when the server response is ready
* Send the request off to a PHP file (gethint.php) on the server
* Notice that q parameter is added gethint.php?q="+str
* The str variable holds the content of the input field

## The PHP File - "gethint.php"

The PHP file checks an array of names, and returns the corresponding name(s) to the browser:

<?php  
// Array with names  
$a[] = "Anna";  
$a[] = "Brittany";  
$a[] = "Cinderella";  
$a[] = "Diana";  
$a[] = "Eva";  
$a[] = "Fiona";  
$a[] = "Gunda";  
$a[] = "Hege";  
$a[] = "Inga";  
$a[] = "Johanna";  
$a[] = "Kitty";  
$a[] = "Linda";  
$a[] = "Nina";  
$a[] = "Ophelia";  
$a[] = "Petunia";  
$a[] = "Amanda";  
$a[] = "Raquel";  
$a[] = "Cindy";  
$a[] = "Doris";  
$a[] = "Eve";  
$a[] = "Evita";  
$a[] = "Sunniva";  
$a[] = "Tove";  
$a[] = "Unni";  
$a[] = "Violet";  
$a[] = "Liza";  
$a[] = "Elizabeth";  
$a[] = "Ellen";  
$a[] = "Wenche";  
$a[] = "Vicky";  
  
// get the q parameter from URL  
$q = $\_REQUEST["q"];  
  
$hint = "";  
  
// lookup all hints from array if $q is different from ""  
if ($q !== "") {  
  $q = strtolower($q);  
  $len=strlen($q);  
  foreach($a as $name) {  
    if (stristr($q, substr($name, 0, $len))) {  
      if ($hint === "") {  
        $hint = $name;  
      } else {  
        $hint .= ", $name";  
      }  
    }  
  }  
}  
  
// Output "no suggestion" if no hint was found or output correct values  
echo $hint === "" ? "no suggestion" : $hint;  
?>

# **XML Applications**

<!DOCTYPE html>

<html>

<style>

table,th,td {

border : 1px solid black;

border-collapse: collapse;

}

th,td {

padding: 5px;

}

</style>

<body>

<button type="button" onclick="loadXMLDoc()">Get my CD collection</button>

<br><br>

<table id="demo"></table>

<script>

function loadXMLDoc() {

var xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

myFunction(this);

}

};

xmlhttp.open("GET", "cd\_catalog.xml", true);

xmlhttp.send();

}

function myFunction(xml) {

var i;

var xmlDoc = xml.responseXML;

var table="<tr><th>Artist</th><th>Title</th></tr>";

var x = xmlDoc.getElementsByTagName("CD");

for (i = 0; i <x.length; i++) {

table += "<tr><td>" +

x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +

"</td><td>" +

x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +

"</td></tr>";

}

document.getElementById("demo").innerHTML = table;

}

</script>

</body>

</html>