What is AJAX?

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

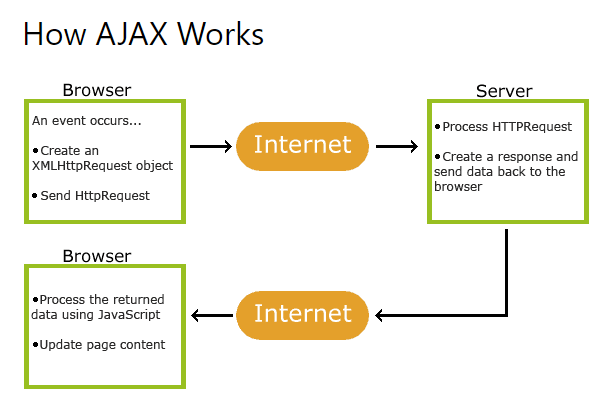
AJAX is not a programming language.

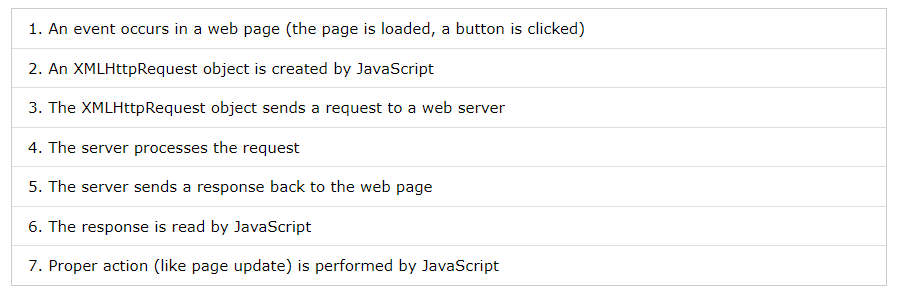
AJAX just uses a combination of:

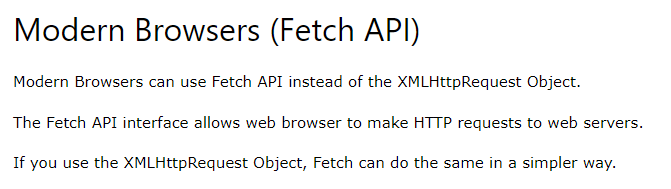
* A browser built-in XMLHttpRequest object (to request data from a web server)
* JavaScript and HTML DOM (to display or use the data)

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.







## Create an XMLHttpRequest Object

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

Syntax for creating an XMLHttpRequest object:

*variable*= new XMLHttpRequest();

## Define a Callback Function

A callback function is a function passed as a parameter to another function.

In this case, the callback function should contain the code to execute when the response is ready.

xhttp.onload = function() {  
  // What to do when the response is ready  
}

## Send a Request

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

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

## Access Across Domains

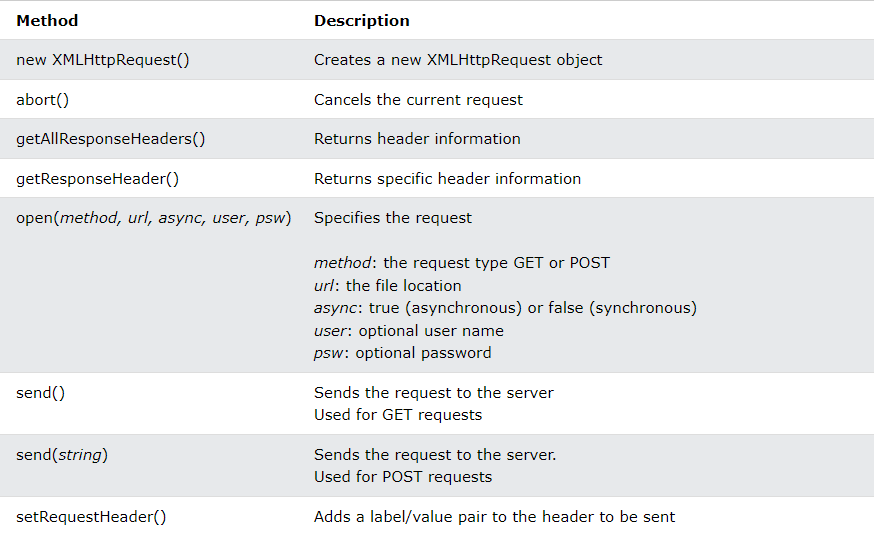
For security reasons, modern browsers do not allow access across domains.

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

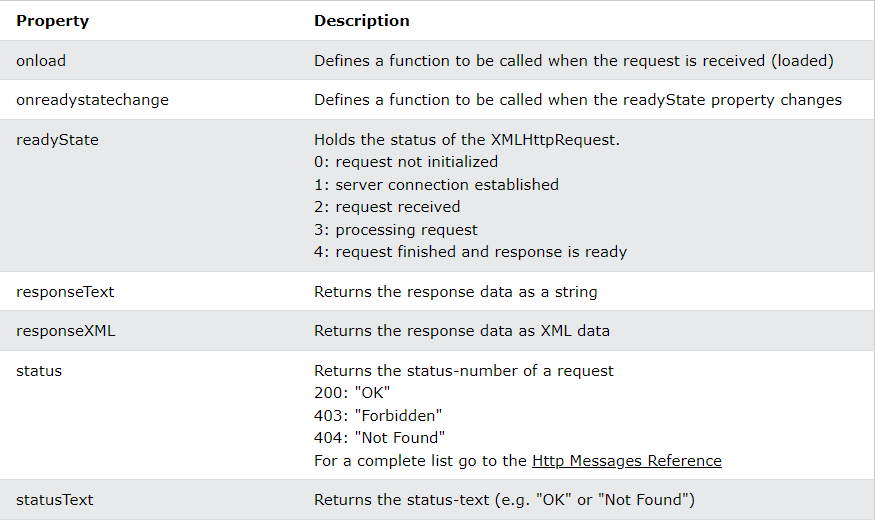
The examples on W3Schools all open XML files located on the W3Schools domain.

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.

## XMLHttpRequest Object Methods



## XMLHttpRequest Object Properties



## The onload Property

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

## Multiple Callback Functions

If you have more than one AJAX task in a website, you should create one function for executing the XMLHttpRequest object, and one callback function for each AJAX task.

## The onreadystatechange Property

The readyState property holds the status of the XMLHttpRequest.

The onreadystatechange property defines a callback function to be executed when the readyState changes.

The status property and the statusText properties hold the status of the XMLHttpRequest object.

The onreadystatechange function is called every time the readyState changes.

When readyState is 4 and status is 200, the response is ready:

function loadDoc() {  
  const 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");  
  xhttp.send();  
}



## 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);

## 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);

## 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:

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

## POST Requests

A simple POST request:

xhttp.open("POST", "demo\_post.asp");  
xhttp.send();

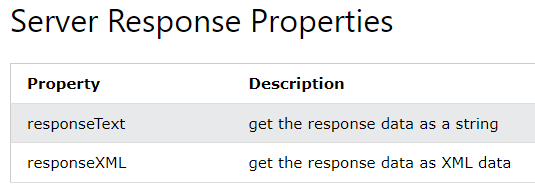
To POST data like an HTML form, add an HTTP header with setRequestHeader(). Specify the data you want to send in the send() method:

xhttp.open("POST", "ajax\_test.asp");  
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");  
xhttp.send("fname=Henry&lname=Ford");

## Synchronous Request

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

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



## The responseText Property

The responseText property returns the server response as a JavaScript string, and you can use it accordingly:

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

## The responseXML Property

The XMLHttpRequest object has an in-built XML parser.

The responseXML property returns the server response as an XML DOM object.

Using this property you can parse the response as an XML DOM object:

## const xmlDoc = xhttp.responseXML; const x = xmlDoc.getElementsByTagName("ARTIST"); let txt = ""; for (let i = 0; i < x.length; i++) {   txt += x[i].childNodes[0].nodeValue + "<br>"; } document.getElementById("demo").innerHTML = txt; xhttp.open("GET", "cd\_catalog.xml"); xhttp.send();

## Server Response Methods

## The getAllResponseHeaders() Method

The getAllResponseHeaders() method returns all header information from the server response.

const xhttp = new XMLHttpRequest();  
xhttp.onload = function() {  
    document.getElementById("demo").innerHTML =  
    this.getAllResponseHeaders();  
}  
xhttp.open("GET", "ajax\_info.txt");  
xhttp.send();

## The getResponseHeader() Method

The getResponseHeader() method returns specific header information from the server response.

const xhttp = new XMLHttpRequest();  
xhttp.onload = function() {  
    document.getElementById("demo").innerHTML =  
    this.getResponseHeader("Last-Modified");  
}  
xhttp.open("GET", "ajax\_info.txt");  
xhttp.send();

What is Difference between get, put, post and delete?

The primary or most-commonly-used HTTP verbs (or methods, as they are properly called) are POST, GET, PUT, and DELETE. These correspond to create, read, update, and delete (or CRUD) operations, respectively. There are a number of other verbs, too, but are utilized less frequently.

Below is a table summarizing the methods used by this service API.

