**WHAT IS AJAX?**

AJAX = Asynchronous JavaScript And XML.

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 WorksWorks

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

The XMLHttpRequest Object

The keystone of AJAX is 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 have a built-in XMLHttpRequest object.

Syntax for creating an XMLHttpRequest object:

variable = new XMLHttpRequest();

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(header, value) Adds HTTP headers to the request

header: specifies the header name

value: specifies the header value

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"

statusText Returns the status-text

(e.g. "OK" or "Not Found")

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();

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", true);

xhttp.send();

In the example above, you may get a cached result. To avoid this, add a unique ID to the URL:

xhttp.open("GET", "demo\_get.asp?t=" + Math.random(), true);

xhttp.send();

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

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

xhttp.send();

POST Requests

A simple POST request:

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

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", true);

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

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

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 XMLHttpRequest object:

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();

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:

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

xhttp.send();

document.getElementById("demo").innerHTML =

xhttp.responseText;

Using a Callback Function

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

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 function call should contain the URL and what function to call when the response is ready.

Example

loadDoc("url-1", myFunction1);

loadDoc("url-2", myFunction2);

function loadDoc(url, cFunction) {

var xhttp;

xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

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

cFunction(this);

}

};

xhttp.open("GET", url, true);

xhttp.send();

}

function myFunction1(xhttp) {

// action goes here

}

function myFunction2(xhttp) {

// action goes here

}

Libraries & Other Methods

There are many different ways to make AJAX request

* jQuery
* Axios
* Superagent
* FetchAPI
* Prototype
* NodeHttp