XML HTTPREQUEST

* All modern browsers have a built-in XMLHttpRequest object to request data from a server.

The XMLHttpRequest Object

The XMLHttpRequest object can be used to request data from a web server.

The XMLHttpRequest object is **a developers dream**, because you can:

* Update a web page without reloading the page
* Request data from a server - after the page has loaded
* Receive data from a server  - after the page has loaded
* Send data to a server - in the background

AJAX

* AJAX is not a programming language.
* AJAX is a technique for accessing web servers from a web page.
* AJAX stands for Asynchronous JavaScript And XML.

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

# AJAX - 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 server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

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() | 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" |
| 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 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.

Example:

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

## POST Requests

A simple POST request:

### Example

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:

|  |  |
| --- | --- |
| **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.

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.

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 - Server Response

## The onreadystatechange Property

The **readyState** property holds the status of the XMLHttpRequest.

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

The **status** property and the **statusText** property holds the status of the XMLHttpRequest object.

|  |  |
| --- | --- |
| **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 |
| status | 200: "OK" 403: "Forbidden" 404: "Page not found" |
| statusText | Returns the status-text (e.g. "OK" or "Not Found") |

The onreadystatechange function is called every time the readyState changes.

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

### Example

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();  
}

The onreadystatechange event is triggered four times (1-4), one time for each change in the readyState.

Below is a list of HTTP status messages that might be returned:

1xx: Information

|  |  |  |  |
| --- | --- | --- | --- |
| **Message:** | | **Description:** | |
| 100 Continue | | The server has received the request headers, and the client should proceed to send the request body |
| 101 Switching Protocols | | The requester has asked the server to switch protocols | |
| 103 Checkpoint | Used in the resumable requests proposal to resume aborted PUT  or POST requests | | |

2xx: Successful

|  |  |
| --- | --- |
| **Message:** | **Description:** |
| 200 OK | The request is OK (this is the standard response for  successful HTTP requests) |
| 201 Created | The request has been fulfilled, and a new resource is created |
| 202 Accepted | The request has been accepted for processing, but the  processing has not been completed |
| 203 Non-Authoritative Information | The request has been successfully processed,  but is returning information that may be from another source |
| 204 No Content | The request has been successfully processed, but is not  returning any content |
| 205 Reset Content | The request has been successfully processed, but is not  returning any content, and requires that the requester  reset the document view |
| 206 Partial Content | The server is delivering only part of the resource due to a  range header sent by the client |

3xx: Redirection

|  |  |  |
| --- | --- | --- |
| **Message:** | | **Description:** |
| 300 Multiple Choices | A link list. The user can select a link and go to that location.  Maximum five addresses | |
| 301 Moved Permanently | The requested page has moved to a new URL | |
| 302 Found | The requested page has moved temporarily to a new URL | |
| 303 See Other | The requested page can be found under a different URL | |
| 304 Not Modified | Indicates the requested page has not been modified  since last requested | |
| 306 Switch Proxy | *No longer used* | |
| 307 Temporary Redirect | The requested page has moved temporarily to a new URL | |
| 308 Resume Incomplete | Used in the resumable requests proposal to resume  aborted PUT or POST requests | |

4xx: Client Error

|  |  |
| --- | --- |
| **Message:** | **Description:** |
| 400 Bad Request | The request cannot be fulfilled due to bad syntax |
| 401 Unauthorized | The request was a legal request, but the server is refusing to  respond to it. For use when authentication is possible but  has failed or not yet been provided |
| 402 Payment Required | *Reserved for future use* |
| 403 Forbidden | The request was a legal request, but the server is  refusing to respond to it |
| 404 Not Found | The requested page could not be found but may be  available again in the future |
| 405 Method Not Allowed | A request was made of a page using a request  method not supported by that page |
| 406 Not Acceptable | The server can only generate a response that is not  accepted by the client |
| 407 Proxy Authentication Required | The client must first authenticate itself with the proxy |
| 408 Request Timeout | The server timed out waiting for the request |
| 409 Conflict | The request could not be completed because of a  conflict in the request |
| 410 Gone | The requested page is no longer available |
| 411 Length Required | The "Content-Length" is not defined. The server will  not accept the request without it |
| 412 Precondition Failed | The precondition given in the request evaluated to  false by the server |
| 413 Request Entity Too Large | The server will not accept the request, because the  request entity is too large |
| 414 Request-URI Too Long | The server will not accept the request, because the URL  is too long. Occurs when you convert a POST request  to a GET request with a long query information |
| 415 Unsupported Media Type | The server will not accept the request, because the  media type is not supported |
| 416 Requested Range Not Satisfiable | The client has asked for a portion of the file,  but the server cannot supply that portion |
| 417 Expectation Failed | The server cannot meet the requirements of  the Expect request-header field |

5xx: Server Error

|  |  |  |  |
| --- | --- | --- | --- |
| **Message:** | | | **Description:** |
| 500 Internal Server Error | A generic error message, given when no more specific message  is suitable | | |
| 501 Not Implemented | | The server either does not recognize the request method, or it  lacks the ability to fulfill the request | |
| 502 Bad Gateway | | The server was acting as a gateway or proxy and received an  invalid response from the upstream server | |
| 503 Service Unavailable | | The server is currently unavailable (overloaded or down) | |
| 504 Gateway Timeout | | The server was acting as a gateway or proxy and did not  receive a timely response from the upstream server | |
| 505 HTTP Version Not Supported | | The server does not support the HTTP protocol version  used in the request | |
| 511 Network Authentication Required | | The client needs to authenticate to gain network access | |