1- XMLHttpRequest

Interface XMLHttpRequest in Javascript is designed to read data source from an URL. Its name can cause confusion that it can read only text/xml data sources. In fact, it can read everything from an URL.

XMLHttpRequest is designed to read the data source from URL synchronously or asynchronously. Reading data asynchronously helps users are still able to manipulate with the browser during the XMLHttpRequest is reading data source remotely.

*If you want to read files on user's computer, the FileReader is the thing you need. It is designed as similarly as XMLHttpRequest.*

* [*Javascript FileReader*](https://o7planning.org/12333/javascript-filereader)

Constructor

// Create a XMLHttpRequest object:

**var** xhr = **new** **XMLHttpRequest**();

Properties

|  |  |
| --- | --- |
| Property | Mô tả |
| readyState | Returns a number, the state of the request. |
| timeout | Specify the maximum time to receive a reply, if it is not received within that time, it is considered a failure and automatically terminated. |
| withCredentials | Its value is true or false (Default is false). If true means that this request can use cookies, authorization headers, but it must still comply with the same origin policy. |
|  | |
| responseType | Specify the type of data you want to receive. The default value is "text". (See possible values of this property below) |
| response | Returns an ArrayBuffer, Blob, Document object, or a DOM String, depending on the value of XMLHttpRequest.responseType, that contains the response entity body. |
| responseText | Returns a DOM String that contains the response to the request as text, or null if the request was unsuccessful or has not yet been sent. |
| responseURL | Returns the serialized URL of the response or the empty string if the URL is null. |
| responseXML | Returns a Document containing the response to the request, or null if the request was unsuccessful, has not yet been sent, or cannot be parsed as XML or HTML. |
|  | |
| status | Returns a number with the status of the response of the request. |
| statusText | Returns a DOM String containing the response string returned by the HTTP Server. For example "200 OK". |

readyState

Possible values of readyState:

|  |  |  |
| --- | --- | --- |
| State | Value | Description |
| UNSENT | 0 | XMLHttpRequest has been created. open() not called yet. |
| OPENED | 1 | open() has been called. |
| HEADERS\_RECEIVED | 2 | send() has been called, and headers & status are available. |
| LOADING | 3 | Downloading; responseText holds partial data. |
| DONE | 4 | The operation is complete. |

responseType

Possible values of responseType:

| **Value** | **Description** |
| --- | --- |
| "" | If no value is specified for the responseType, or empty value, it is treated as "text". |
| "arraybuffer" | The response is a ArrayBuffer containing binary data. |
| "blob" | The response is a Blob object containing the binary data. |
| "document" | The response is an HTML Document or XMLDocument, as appropriate based on the MIME type of the received data. |
| "json" | The response is a JavaScript object created by parsing the contents of received data as JSON. |
| "text" | The response is text in a DOMString object. |

Events

During sending a request for reading a data source from an URL, XMLHttpRequest will fire events described in interface ProgressEvent.

Note: The events marked by (?) are testing standard, which are not supported by most of browsers.

|  |  |  |
| --- | --- | --- |
|  | **Event** | **Description** |
| ? | loadstart | Indicates that the process of loading data has begun. This event always fires first. |
|  | progress | Event fires multiple times as data is being loaded, giving access to intermediate data. |
| ? | error | Event fires when loading has failed. |
| ? | abort | Event fires when data loading has been canceled by calling abort() method (Method available on both XMLHttpRequest & FileReader). |
|  | load | Event fires only when all data has been successfully read. |
| ? | loadend | Event fires when the object has finished transferring data. Always fires after error, abort, or load. |

* *TODO Link?*

Handlers

Note: The handlers marked by (?) are a testing standard, which is not supported by most browsers.

|  | **Handler** | **Mô tả** |
| --- | --- | --- |
|  | onreadystatechange(event) | A handler, which is called when the readyState property changes. |
|  |  |  |
| ? | onloadstart(progressEvt) | A handler for the loadstart event. |
|  | onprogress(progressEvt) | A handler for the progress event. |
| ? | onerror(progressEvt) | A handler for the error event. |
| ? | onabort(progressEvt) | A handler for the abort event. |
|  | onload(progressEvt) | A handler for the load event. |
| ? | onloadend(progressEvt) | A handler for the loadend event. |

Example

<!DOCTYPE **html**>

<**html**>

<**head**>

<**title**>XMLHttpRequest Example</**title**>

<**meta** charset="UTF-8">

<**script** src="xhr-example.js"></**script**>

</**head**>

<**body**>

<**h3**>XMLHttpRequest example</**h3**>

<**a** href="">Reset</**a**> <**br**><**br**>

<**button** onclick = "clickHandler(event)">Click Me</**button**>

<**br**><**br**>

<**textarea** id="textarea-log" cols="50" rows="15"></**textarea**>

</**body**>

</**html**

Script

**function** **clickHandler**(evt) {

**var** URL= "https://ex1.o7planning.com/\_testdatas\_/simple-xml-data.xml";

// var URL= "https://ex1.o7planning.com/\_testdatas\_/triceratops.png";

**resetLog**();

// Create XMLHttpRequest.

**let** xhr = **new** **XMLHttpRequest**();

**appendLog**("URL: " + URL);

**appendLog**("\n\n");

xhr.onprogress = **function**(progressEvent) {

**appendLog**("onprogress! " + progressEvent);

}

// readyState (State of request):

// 0 - XMLHttpRequest.UNSENT

// 1 - XMLHttpRequest.OPENED

// 2 - XMLHttpRequest.HEADERS\_RECEIVED

// 3 - XMLHttpRequest.LOADING

// 4 - XMLHttpRequest.DONE

xhr.onreadystatechange = **function**(event) {

**appendLog**("onreadystatechange! readyState = " + xhr.readyState);

**appendLog**(" status = " + xhr.status);

**appendLog**(" statusText = " + xhr.statusText);

}

xhr.onload = **function**(progressEvent) {

**appendLog**("onload!");

**appendLog**(" status = " + xhr.status);

**appendLog**(" statusText = " + xhr.statusText);

**appendLog**(" ------ xhr.responseText ------: ");

**appendLog**(xhr.responseText);

**appendLog**(" ------ xhr.responseXML -------: ");

**appendLog**(xhr.responseXML); // [object XMLDocument]

// Convert XMLDocument to String.

**var** xmlString = (**new** **XMLSerializer**()).**serializeToString**(xhr.responseXML);

**appendLog**(xmlString);

}

xhr.onerror = **function**(progressEvent) {

**appendLog**("onerror!");

**appendLog**("Has Error!");

}

**let** **async** = true;

// Initialize It.

xhr.**open**("GET", URL, **async**);

// Send it (Without body data)

xhr.**send**();

}

**function** **resetLog**() {

document.**getElementById**('textarea-log').value = "";

}

**function** **appendLog**(msg) {

document.**getElementById**('textarea-log').value += "\n" + msg;

}