AJAX - The XMLHttpRequest Object

‹ Previous Next ›

The keystone of AJAX is the XMLHttpRequest object.

1. Create an XMLHttpRequest object
2. Define a callback function
3. Open the XMLHttpRequest object
4. Send a Request to a server

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, 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();
```

Example

```
// Create an XMLHttpRequest object
const xhttp = new XMLHttpRequest();

// Define a callback function
xhttp.onload = function() {
  // Here you can use the Data
}

// Send a request
xhttp.open("GET", "ajax_info.txt");
xhttp.send();
```

Try it Yourself »

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

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
onload	Defines a function to be called when the request is received (loaded)
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
statusText	Returns the status-text (e.g. "OK" or "Not Found")

The onload Property

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

The function is defined in the `onload` property of the `XMLHttpRequest` object:

Example

```
xhttp.onload = function() {
  document.getElementById("demo").innerHTML = this.responseText;
}
```

```
xhttp.open("GET", "ajax_info.txt");
xhttp.send();
```

Try it Yourself »

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 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) {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {cFunction(this);}
  xhttp.open("GET", url);
  xhttp.send();
}

function myFunction1(xhttp) {
  // action goes here
}
function myFunction2(xhttp) {
  // action goes here
}
```

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.

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" For a complete list go to the Http Messages Reference
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() {
  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();
}
```

Try it Yourself »

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

‹ Previous

Next ›

COLOR PICKER

SPACES

UPGRADE

AD-FREE

NEWSLETTER

GET CERTIFIED

CONTACT US

Top Tutorials

HTML Tutorial

CSS Tutorial

Top References

Top Examples

Get Certified

SQL Certificate
Python Certificate
PHP Certificate
jQuery Certificate
Java Certificate
C++ Certificate
C# Certificate
XML Certificate

FORUM ABOUT CLASSROOM