


**AJAX** |

# THE USUAL WAY WE OPERATE IN THE WEB

Typical browsing behaviour consists of loading a web page, then selecting some action that we want to do, filling out a form, submitting the information, etc.

We work in this sequential manner, requesting one page at a time, and have to wait for the server to respond, loading a whole new web page before we continue.

This is also one of the limitations of web pages, where transmitting information between a client and server generally requires a new page to be loaded.



JavaScript is one way to cut down on (some of) the client-server response time, by using it to verify form (or other) information before it's submitted to a server.

One of the limitations of JavaScript is (or used to be) that there was no way to communicate directly with a web server.

# THINGS CHANGE...

We used to not have any alternative to this load/wait/respond method of web browsing.

Ajax (sometimes written AJAX) is a means of using JavaScript to communicate with a web server without submitting a form or loading a new page.

Ajax makes use of a built-in object, XMLHttpRequest, to perform this function.

# AJAX

Ajax stands for “Asynchronous JavaScript and XML”.

The word “asynchronous” means that the user isn’t left waiting for the server to respond to a request, but can continue using the web page.

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 applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.

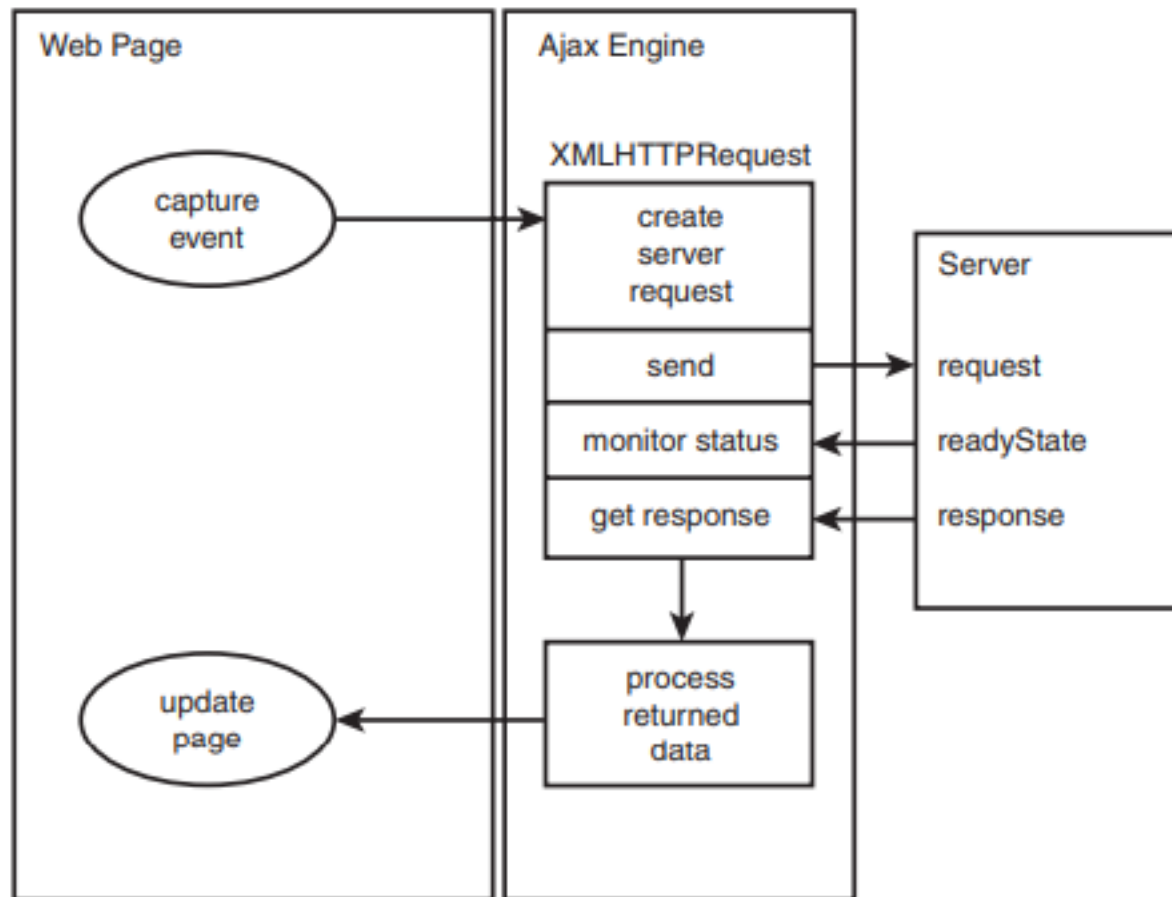
AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

# AJAX

The typical method for using Ajax is the following:

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

## Anatomy of an Ajax Application



# THE BACK-END

The part of the Ajax application that resides on the web server is referred to as the “back-end”.

This back end could be simply a file that the server passes back to the client, which is then displayed for the user.

Alternatively, the back-end could be a program, written in PHP, Perl, Ruby, Python, C, or some other language that performs an operation and sends results back to the client browser.

An XMLHttpRequest object can send information using the GET and POST methods to the server in the same way that an HTML form sends information.



# THE XMLHttpRequest OBJECT

The XMLHttpRequest object is the backbone of every Ajax method.

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

Syntax for creating an XMLHttpRequest object:

```
<script>  
    variable = new XMLHttpRequest();  
</script>
```

# THE XMLHTTPREQUEST OBJECT (CONT.)

As with any object in JavaScript (and other programming languages), the XMLHttpRequest object contains various properties and methods.

The properties are set after the object is created to specify information to be sent to the server, as well as how to handle the response received from the server.

Some properties will be updated to hold status information about whether the request finished successfully.

The methods are used to send the request to the server, and to monitor the progress of the request as it is executed (and to determine if it was completed successfully).

# XMLHttpRequest OBJECT PROPERTIES

| Property           | Description  |
|--------------------|--|
| onreadystatechange | Defines a function to be called when the readyState property changes   |
| readyState         | Holds the status of the XMLHttpRequest.<br>0: request not initialized<br>1: server connection established<br>2: request received<br>3: processing request<br>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<br>200: "OK"<br>403: "Forbidden"<br>404: "Not Found"<br>For a complete list go to the <a href="#">Http Messages Reference</a>                                   |
| statusText         | Returns the status-text (e.g. "OK" or "Not Found")   |

# XMLHttpRequest OBJECT METHODS

| Method  | Description  |
|---|--|
| <code>new XMLHttpRequest()</code>   | Creates a new XMLHttpRequest object  |
| <code>abort()</code>  | Cancels the current request  |
| <code>getAllResponseHeaders()</code>  | Returns header information   |
| <code>getResponseHeader()</code>  | Returns specific header information  |
| <code>open(<i>method</i>,<i>url</i>,<i>async</i>,<i>user</i>,<i>psw</i>)</code> | Specifies the request<br><br><i>method</i> : the request type GET or POST<br><i>url</i> : the file location<br><i>async</i> : true (asynchronous) or false (synchronous)<br><i>user</i> : optional user name<br><i>psw</i> : optional password |
| <code>send()</code>   | Sends the request to the server<br>Used for GET requests   |
| <code>send(<i>string</i>)</code>  | Sends the request to the server.<br>Used for POST requests   |
| <code>setRequestHeader()</code>   | Adds a label/value pair to the header to be sent   |

# EXAMPLE

```
<html>
<body>
```

```
<h2>The XMLHttpRequest Object</h2>
```

```
<div id="demo">
```

```
<button type="button" onclick="loadDoc()">Change Content</button>
```

```
</div>
```

```
<script>
```

```
function loadDoc()
```

```
{
```

```
  const xhttp = new XMLHttpRequest();
```

```
  xhttp.onload = function() {
```

```
    document.getElementById("demo").innerHTML = this.responseText;
```

```
  }
```

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

```
  xhttp.send();
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

The XMLHttpRequest Object

Change Content

The XMLHttpRequest Object

## AJAX

AJAX is not a programming language.

# EXAMPLE

```
<!DOCTYPE html>
<body>
```

```
<div id="demo">
<h1>The XMLHttpRequest Object</h1>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>
```

```
<script>
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();
}
</script>
```

```
</body>
</html>
```

The XMLHttpRequest Object

Change Content

The XMLHttpRequest Object

## AJAX

AJAX is not a programming language.

# GET REQUESTS

`open(method, url, async)`

A simple GET request:

```
xhttp.open("GET", "demo_get.asp");
```

```
xhttp.send();
```

```
<html>
<body>
```

```
<h2>The XMLHttpRequest Object</h2>
```

```
<button type="button" onclick="loadDoc()">Request data</button>
```

```
<p id="demo"></p>
```

```
<script>
```

```
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML = this.responseText;
  }
  xhttp.open("GET", "demo_get2.asp?fname=Henry&lname=Ford");
  xhttp.send();
}
</script>
```

```
</body>
```

```
</html>
```

The XMLHttpRequest Object

Request data

The XMLHttpRequest Object

Request data

Hello Henry Ford



# POST REQUESTS

`open(method, url, async)`

A simple POST request:

Example

```
xhttp.open("POST", "demo_post.asp");
```

```
xhttp.send(data);
```

```
<html>
<body>
```

```
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Request data</button>
```

```
<p id="demo"></p>
```

```
<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML = this.responseText;
  }
  xhttp.open("POST", "demo_post2.asp");
  xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  xhttp.send("fname=Henry&lname=Ford");
}
</script>

</body>
</html>
```

The XMLHttpRequest Object

Request data

The XMLHttpRequest Object

Request data

Hello Henry Ford

# SYNCHRONOUS REQUEST

By default, it is true (async).

You can also set it with the third parameter of `open()` method to either true or false.

Example

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

```
xhttp.send();
```

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

```
<html>
<body>
```

```
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Request
data</button>
```

```
<p id="demo"></p>
```

```
<script>
function loadDoc() {
    var xhttp = new XMLHttpRequest();
    xhttp.open("GET", "ajax_info.txt", false);
    xhttp.send();
    document.getElementById("demo").innerHTML =
xhttp.responseText;
}
</script>
```

```
</body>
</html>
```

## The XMLHttpRequest Object

Request data

## The XMLHttpRequest Object

Request data

Hello Henry Ford

# SERVER RESPONSE

## **The `responseText` Property**

The `responseText` property returns the server response as a JavaScript string.

## **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

```
<html>
<body>
```

```
<div id="demo">
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Change
Content</button>
</div>
```

```
<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML =
    this.responseText;
  }
  xhttp.open("GET", "ajax_info.txt");
  xhttp.send();
}
</script>
</body>
</html>
```

## The XMLHttpRequest Object

Change Content

# AJAX

AJAX is not a programming language.

```
<html>
<body>
<h3>The XMLHttpRequest Object</h3>
<p id="demo"></p>
<script>
const xhttp = new XMLHttpRequest();
xhttp.onload = function() {
  const xmlDoc = this.responseXML;
  const x = xmlDoc.getElementsByTagName("ARTIST");
  let txt = "";
  for (let i = 0; i < x.length; i++) {
    txt = txt + x[i].childNodes[0].nodeValue + "<br>";
  }
  document.getElementById("demo").innerHTML = txt;
}
xhttp.open("GET", "cd_catalog.xml");
xhttp.send();
</script>
</body>
</html>
```

## The XMLHttpRequest Object

Bob Dylan  
Bonnie Tyler  
Dolly Parton

# SERVER RESPONSE METHODS

## **The getAllResponseHeaders() Method**

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

## **The getResponseHeader() Method**

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



**<!DOCTYPE html>**

**<html>**

**<body>**

## The XMLHttpRequest Object

accept-ranges: bytes

age: 83283

cache-control: public,max-age=31536000,public

content-encoding: gzip

content-length: 147

**<h3>The XMLHttpRequest Object</h3>**

**<pre id="demo"></pre>**

**<script>**

**const xhttp = new XMLHttpRequest();**

**xhttp.onload = function() {**

**document.getElementById("demo").innerHTML =**

**this.getAllResponseHeaders();**

**}**

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

**xhttp.send();**

**</script>**

**</body>**

**</html>**

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h3>The XMLHttpRequest Object</h3>
```

```
<pre id="demo"></pre>
```

```
<script>
```

```
const xhttp = new XMLHttpRequest();
```

```
xhttp.onload = function() {
```

```
    document.getElementById("demo").innerHTML =
```

```
    this.getResponseHeader("Content-Length");
```

```
}
```

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

```
xhttp.send();
```

```
</script>
```

```
</body>
```

```
</html>
```

# WITH JQUERY

jQuery provides several methods for AJAX functionality.

With the jQuery AJAX methods, you can request text, HTML, XML, or JSON from a remote server using both HTTP Get and HTTP Post - And you can load the external data directly into the selected HTML elements of your web page!

## **Without jQuery, AJAX coding can be a bit tricky!**

Writing regular AJAX code can be a bit tricky, because different browsers have different syntax for AJAX implementation. This means that you will have to write extra code to test for different browsers. However, the jQuery team has taken care of this for us, so that we can write AJAX functionality with only one single line of code.

# JQUERY LOAD() METHOD

The jQuery load() method is a simple, but powerful AJAX method.

The load() method loads data from a server and puts the returned data into the selected element.

Syntax:

```
$(selector).load(URL,data,callback);
```

The required URL parameter specifies the URL you wish to load.

The optional data parameter specifies a set of querystring key/value pairs to send along with the request.

The optional callback parameter is the function's name to be executed after the load() method is completed.

```
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></
script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#div1").load("demo_test.txt");
    });
});
</script>
</head>
<body>
<div id="div1"><h3>jQuery Load()</h3></div>
<button>Get External Content</button>
</body>
</html>
```

### jQuery Load()

Get External Content

### jQuery and AJAX is FUN!

This is some text in a paragraph.

Get External Content

# JQUERY - AJAX GET() AND POST() METHODS

**GET** is basically used for just getting (retrieving) some data from the server. Note: The GET method may return cached data.

```
$.get(URL, callback);
```

**POST** can also be used to get some data from the server. However, the POST method NEVER caches data, and is often used to send data along with the request.

```
$.post(URL, data, callback);
```

```
<html> <head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js">
</script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $.get("demo_test.asp", function(data, status){
            alert("Data: " + data + "\nStatus: " + status);
        });
    });
});
</script>
</head>
<body>
<button>Send GET request</button>
</body>
</html>
```

Send GET request

Data: This is some text from an external ASP file.  
Status: success

OK

```
<script>
```

```
$(document).ready(function(){
```

```
  $("button").click(function(){
```

```
    $.post("demo_test_post.asp",
```

```
    {
```

```
      name: "Donald Duck",
```

```
      city: "Duckburg"
```

```
    },
```

```
    function(data,status){
```

```
      alert("Data: " + data + "\nStatus: " + status);
```

```
    }); });
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<button>Send HTTP POST request</button>
```

```
</body>
```

```
</html>
```

Send HTTP POST request

Data: Dear Donald Duck. Hope you live well in Duckburg.  
Status: success

OK



# REFERENCE

<http://learn.jquery.com/ajax/>

[https://www.w3schools.com/js/js\\_ajax\\_intro.asp](https://www.w3schools.com/js/js_ajax_intro.asp)

[https://www.w3schools.com/jquery/jquery\\_ajax\\_intro.asp](https://www.w3schools.com/jquery/jquery_ajax_intro.asp)