AJAX is not a programming language.

AJAX is a technique for accessing web page from web server.

AJAX stands for Asynchronous JavaScript And XML.

Using AJAX

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

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)

 The XMLHttpRequest object can be used to exchange data with a web server behind the scenes.

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.

1. Create an XMLHttpRequest object

const xhttp = new XMLHttpRequest();

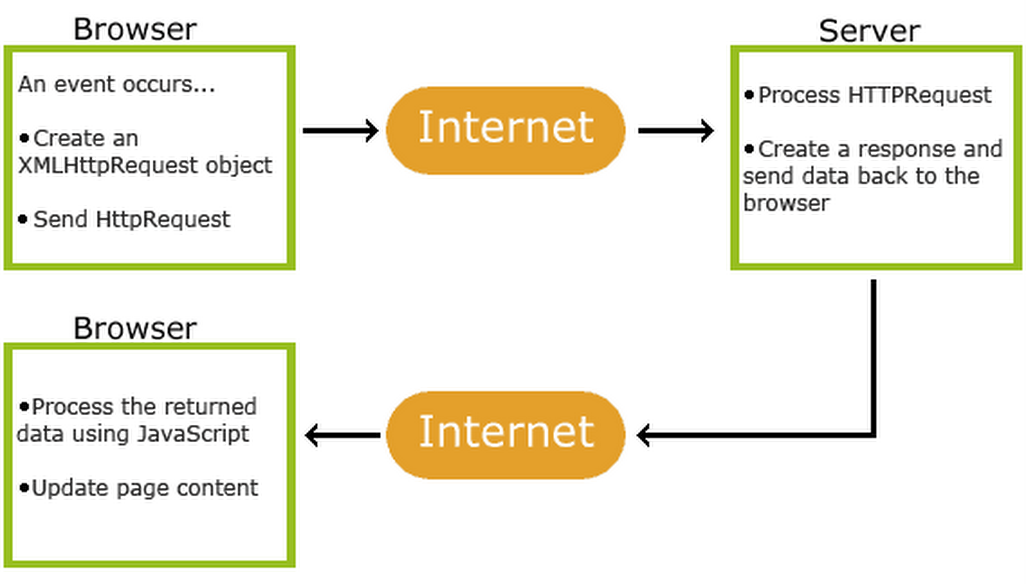
1. Define a callback function

xhttp.onreadystatechange= function() {  
  // Here you can use the Data  
}

1. Open the XMLHttpRequest object
2. Send a Request to a server

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",true);  
xhttp.send();



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
8. XMLHttpRequest Object Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| onload | Defines a function to be called when the request is recieved (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](https://www.w3schools.com/tags/ref_httpmessages.asp) |
| 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:

|  |  |
| --- | --- |
| **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) |

Server requests should be sent asynchronously.

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

The default value for the async parameter is async = true.

GET is simpler and faster than POST, and can be used in most cases.

|  |  |
| --- | --- |
| **Property** | **Description** |
| responseText | get the response data as a string |
| responseXML | get the response data as XML data |

To retrieve data from JSON file using jQuery and Ajax, use the

jQuery.getJSON( url, [data], [callback] )

* **url**− A string containing the URL to which the request is sent
* **data**− This optional parameter represents key/value pairs that will be sent to the server.
* **callback**− This optional parameter represents a function to be executed whenever the data is loaded successfully.

result.json file −

{

"name": "John",

"age" : "30",

"sex": "male"

}

<head>

      <title>Readf Json fil - The jQuery Example</title>

<script>

         $(document).ready(function() {

            $("#btn").click(function(event){

               $.getJSON('result.json', function(jd) {

                $('#stage').html('<p> Name: ' + jd.name + '</p>');

                  $('#stage').append('<p>Age : ' + jd.age+ '</p>');

                  $('#stage').append('<p> Sex: ' + jd.sex+ '</p>');

               });

            });

         });

      </script>

</head>

   <body>

      <div id = "res" style = "background-color:#cc0;">

         Result

      </div>

      <input type = "button" id = "btn" value = "Load Data" />

   </body>

the same request can be triggered with the more verbose $.ajax() call. Here we would use:

$.ajax({

dataType: 'json',

url: url,

data: data,

success: success

});

**Make AJAX Call to Populate HTML Table with JSON Data:**

And this is the jquery script for making ajax call to receive json data over HTTP communication. The script basically sends HTTP request to get json from an url and then parse the received json & display it in the html table we have created in the above step.

The jquery ajax() method makes asynchronous communication between the client and server over HTTP.

Its **URL** parameter specifies the url to which the HTTP request should be sent. In our example it is the path to the json file.

The **dataType** is the expected datatype of the server response.

The **success** parameter contains the function to be called once the AJAX request is successfully completed.

And the **error** parameter contains the function to be executed when the server request fails.