1. What Is Ajax?

Ajax is an acronym for **Asynchronous Javascript and XML**. It is used to communicate with the server without refreshing the web page and thus increasing the user experience and better performance.

- It is a group of inter-related technologies like JavaScript, DOM, XML, HTML/XHTML, CSS, and XMLHttpRequest etc.
- AJAX allows you to send and receive data asynchronously without reloading the web page. So it is fast.
- **AJAX allows you to send only important information to the server not the entire page**. So only valuable data from the client side is routed to the server side. It makes your application **interactive and faster.**
- Ajax uses

  XHTML for CONTENT,

  CSS for PRESENTATION,

  DOM & JavaScript for DYNAMIC CONTENT DISPLAY.
- Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
- XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
- AJAX is a web browser technology independent of web server software.
- A user can continue to use the application while the client program requests information from the server in the background.
- Intuitive and natural user interaction. Clicking is not required; mouse movement is a sufficient event trigger.
- Data-driven as opposed to page-driven

Where It Is Used?

There are too many web applications running on the web that are using Ajax technology like **Gmail**, **Facebook**, **twitter**, **Google map**, **YouTube** etc.

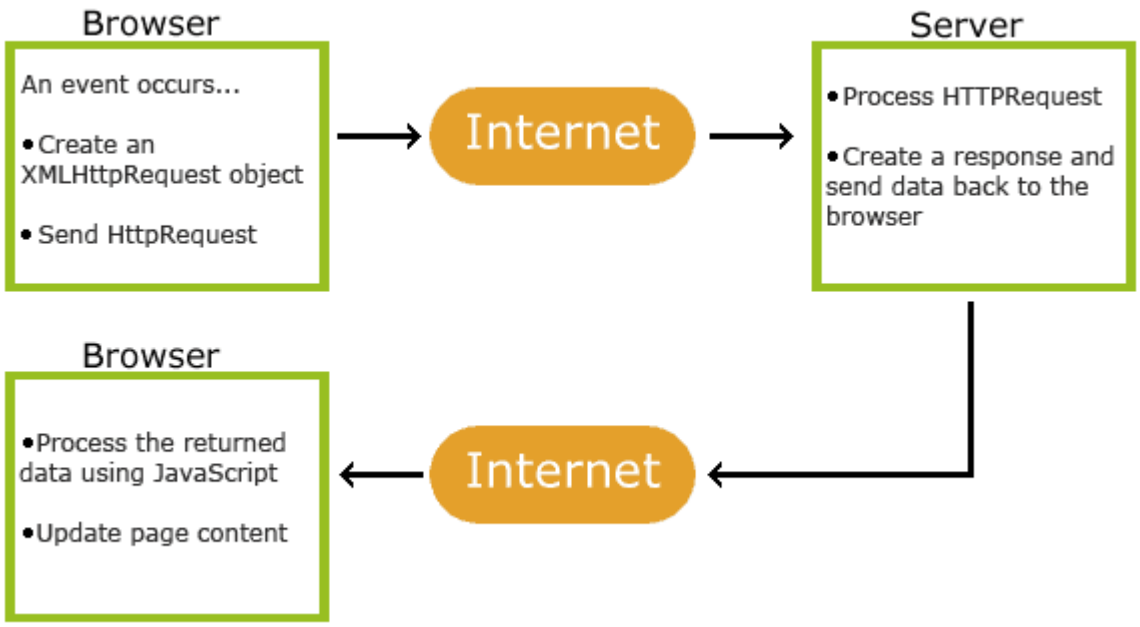
Synchronous (Classic Web-Application Model)

A synchronous request blocks the client until operation completes i.e. browser is unresponsive. In such case, javascript engine of the browser is blocked.

Asynchronous (AJAX Web-Application Model)

An asynchronous request doesn't block the client i.e. browser is responsive. At that time, user can perform other operations also. In such case, javascript engine of the browser is not blocked.

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

XMLHttpRequest

XMLHttpRequest has two modes of operation: synchronous and asynchronous.

Let's see the asynchronous first, as it's used in the majority of cases.

To do the request, we need 3 steps:

1. **Create XMLHttpRequest:**

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

2. **Initialize it, usually right after new XMLHttpRequest:**

   ```
   xhr.open (method, URL, [async, user, password])
   ```

   This method specifies the main parameters of the request:

   method – HTTP-method. Usually "GET" or "POST".

   URL – the URL to request, a string, can be URL object.

   async – if explicitly set to false, then the request is synchronous, we'll cover that a bit later.

   user, password – login and password for basic HTTP auth (if required).

   Please note that open call, contrary to its name, does not open the connection. It only configures the request, but the network activity only starts with the call of send.

3. **Send it out.**

   ```
   xhr.send ([body])
   ```

   This method opens the connection and sends the request to server. The optional body parameter contains the request body.

2. Call HTTP Methods Using AJAX

An object of XMLHttpRequest is used for asynchronous communication between client and server.

It performs following operations:

1. Sends data from the client in the background
2. Receives the data from the server
3. Updates the webpage without reloading it.

HTTP Request: GET vs. POST

Two commonly used methods for a request-response between a client and server are: GET and POST.

GET - Requests data from a specified resource

GET is basically used for just getting (retrieving) some data from the server.

Note: The GET method may return cached data.

POST - Submits data to be processed to a specified resource

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.

Note : POST method NEVER caches data

Methods of XMLHttpRequest object

The important methods of XMLHttpRequest object are as follows:

Method	Description
void open(method, URL)	opens the request specifying get or post method and url.
void open(method, URL, async)	same as above but specifies asynchronous or not.
void open(method, URL, async, username, password)	same as above but specifies username and password.
void send()	sends get request.
void send(string)	send post request.
setRequestHeader(header,value)	it adds request headers.

AJAX error handling

Step 1: Add timeout

The $.ajax method lets you set a timeout in milli seconds.

When a timeout happens,

- The fail callback is called, with errorThrown set to "timeout".
- The request is aborted, meaning that even if the response arrives later on, your done callback is not called by jQuery.

```
var requestData = data to send to server;
var url = Url to send request to;

// Show spinner image

$.ajax(url, {
    "data": requestData,
    "type": "POST",
    "timeout": 5000
})

.done(function (data, textStatus, jqXHR) {
    // Process data, as received in data parameter
})
.fail(function (jqXHR, textStatus, errorThrown) {
    // Request failed. Show error message to user.
    // errorThrown has error message, or "timeout" in case of timeout.
})
.always(function(jqXHR, textStatus, errorThrown) {
    // Hide spinner image
}
```

Step 2: Log fatal message in case of error or timeout

When there is an AJAX error response or the AJAX request times out, you'll want to **log as much information as you have, including the error message** that jQuery gives you, the url and the request data.

```
JL().fatal({
    "msg": "AJAX error response",
    "errorThrown": errorThrown,
    "url": url,
    "requestData": requestData
});
```

Step 3: Log warning message if AJAX response takes longer than expected

Record the time before making the AJAX call and compare that with the time when the response is received to find out how long the user had to wait for the response.

Log a warning message if it took longer than expected.

```
var requestData = data to send to server;
var url = Url to send request to;
// Show spinner image
var msBeforeAjaxCall = new Date().getTime();

$.ajax(url, {
    "data": requestData,
    "type": "POST",
    "timeout": 5000
})
.done(function (data, textStatus, jqXHR) {
    // Process data, as received in data parameter
```

```
// Send warning log message if response took longer than 2 seconds
        var msAfterAjaxCall = new Date().getTime();
        var timeTakenInMs = msAfterAjaxCall - msBeforeAjaxCall;
        if (timeTakenInMs > 2000) {
            JL().warn({
                "msg": "AJAX response took long time",
                "timeTakenInMs": timeTakenInMs,
                "url": url,
                "data": data,
                "requestData": requestData
            });
        }
    })
    .fail(function (jqXHR, textStatus, errorThrown) {
        // Request failed. Show error message to user.
        // errorThrown has error message, or "timeout" in case of
    timeout.

        JL().fatal({
                    "msg": "AJAX error response",
                    "errorThrown": errorThrown,
                    "url": url,
                    "requestData": requestData
                });
    })
    .always(function(jqXHR, textStatus, errorThrown) {
        // Hide spinner image
    }
```