DESCRIPTION

**AJAX**

AJAX stands for **A**synchronous **J**avaScript And **X**ML. Ajax describes the process of exchanging data from a web server asynchronously with the help of XML, HTML, CSS, and JavaScript. It just loads the data from the server and selectively updates some webpage parts without refreshing the page. Ajax uses the browser's built-in XMLHttpRequest (XHR) object to send and receive data to and from a web server asynchronously, in the background, without blocking the page or interfering with the user's experience.

Despite the name, modern usage of AJAX usually works with JSON data rather than XML as we will see.

**Ajax Work Flow**

Ajax works flow starts from the client-side, on the browser. The steps involved in Ajax communication as follows:

1. A client event occurs on a webpage. (for example, the user clicks a button)
2. JavaScript creates an XMLHttpRequest object.
3. The XMLHttpRequest object makes an asynchronous request to the server.
4. The server process the received HttpRequest.
5. The server creates a response and sends data back to the browser.
6. Browser process the returned data using JavaScript.
7. The page content updated by javascript.

**The XMLHttpRequest Object**

The keystone of AJAX is the XMLHttpRequest object. An XMLHttpRequest (XHR) object used to make HTTP requests to the server and receive data in response.

**Create an XMLHttpRequest Object:**

Before you perform Ajax communication between client and server, we should create an XMLHttpRequest object.

var xhttp = new XMLHttpRequest();

**Send a Request To a Server:**

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object.

* **open(method, URL, async)**

where, method — Specifies the HTTP request method to use, such as "GET", "POST", etc., URL - Specifies the location of the server async - Specifies whether the request should be handled asynchronously or not. If "true" then the script processing carries without waiting for a response. If "false" then the script waits for a response.

* **send()** - Sends the request to the server. It accepts an optional parameter that allows us to specify the request's body (used for POST).

In the GET method, the data is sent as URL parameters. In the POST method, the data is sent to the server as a part of the HTTP request body, which is not visible in the URL.

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

xhttp.send();

**Server Response**

The Server Response returned in the form of responseText or responseXML or status or statusText.

* **responseText** - Returns the response as a string.
* **responseXML** - Returns the response as XML.
* **status** - Returns the status as a number (For example, 200: "OK", 403: "Forbidden",404: "Page not found")
* **statusText** -Returns the status as a string (e.g., "Not Found" or "OK").

We have to wait for the data to be available to process it, and in this purpose, the state of availability of data is given by the **readyState** attribute of XMLHttpRequest. The **onreadystatechange** function is called every time the readyState changes.

The readyState property defines the status of the XMLHttpRequest:

* readyState = 0 : Not intialized
* readyState = 1 : Connection establised
* readyState = 2 : request received
* readyState = 3 : processing request
* readyState = 4 : request finished and response is ready

**Example:**

<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>

**Working with JSON in Ajax**

Sending JSON request payload and receiving the JSON response object are common tasks while dealing with AJAX. JavaScript retrieves the JSON data sent by the server, parse them, and displaying them on the webpage.

**JSON in Request Payload**

In JavaScript, to send a request using JSON data, we need to serialize our JSON object into a string. The **JSON.stringify()** method is used to converting an object to a string. Then, the server receives the string and process the request.

**Example:**

var data = {"name" : "Matt"};

var xmlhttp = new XMLHttpRequest();

xmlhttp.open("POST", "/demo", true);

//Use stringify() method to get string

xmlhttp.send( JSON.stringify( data ) );

**JSON in Response Body**

If the response from the server is string/text, we need to parse them into a JSON object. The **JSON.parse()** method converts a JSON string representation to a JSON object.

**Example:**

var xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

//Use parse() method to convert JSON string to JSON object

var responseJsonObj = JSON.parse(this.responseText);

console.log( responseJsonObj.name );

console.log( responseJsonObj.age );

}

};

xmlhttp.open("GET", "/demo", true);

xmlhttp.send();

**XML in Response Body**

While we normally wish to parse our response from JSON, sometimes we might be dealing with an older server and we need to parse XML data. The **responseXML** field contains the data as a read-only Document Object Model.

**Example:**

var xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

var responseXml = this.responseXML;

console.log( responseXml.getElementsByTagName('myTag') )

}

};

xmlhttp.open("GET", "/demo", true);

xmlhttp.send();