AJAX

AJAX is an acronym for **Asynchronous Javascript and XML**. It is a technique for creating better, faster and more interactive web applications with the help of **XML, CSS, and JavaScript**. Unlike classic web pages, where the entire page loads if content changes, AJAX allows web pages to be updated asynchronously by fetching data from the server behind the scenes.

It provides a framework where JavaScript code can communicate with the server in a simple and standard way. AJAX uses plain text, XML, any type of data as the format for receiving server's data.

Many famous web applications now use AJAX technology. For example

- Google Maps
- Gmail
- Google search engine.
- Facebook

AJAX Benefits

- Before AJAX, the client side of a web application could not communicate directly with the server without refreshing the page. AJAX makes this possible. AJAX allows the client and server of a web application to freely communicate with each other.
- The primary purpose of AJAX is to modify the part of the web page already displayed in the browser's window without reloading the entire page. As it loads only the relevant portion instead of the entire page, it is faster than the traditional technologies.
- AJAX allows JavaScript to communicate with the server asynchronously. JavaScript accepts a request from the user and sends it to the server using a separate thread. The user is free to do other things at that time. The response comes back, JavaScript code takes care about the response and displays the result. This provides users a very friendly experience.

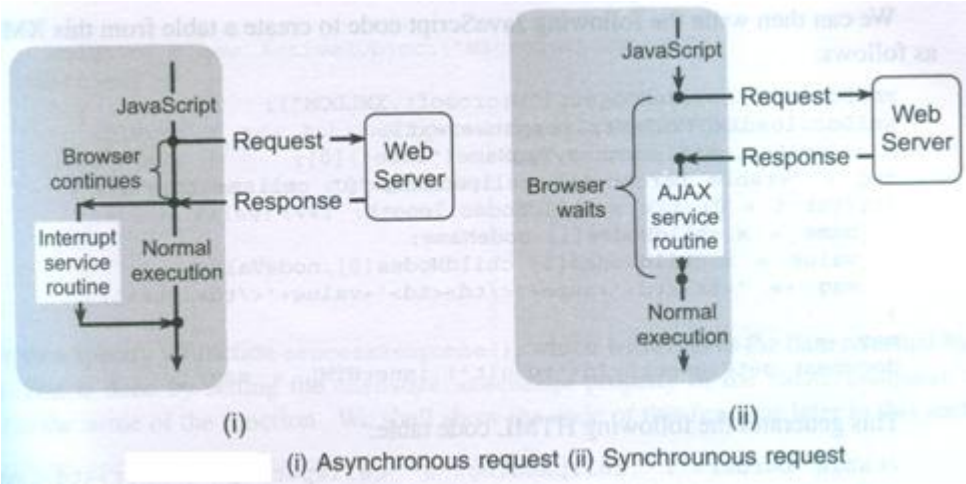
Basic Idea of AJAX

The basic idea of AJAX is that it allows users to fetch data from the web servers asynchronously.

In **HTTP request-response protocol**, a server process runs and listens for incoming requests. To get a service, one can send an HTTP request to the server according to the rules specified by HTTP. The server then accepts, processes and understands the request and sends the response data back to the process that made the request. In this scenario, the client must be blocked entirely till it gets response back. This is called **synchronous communication**.

JavaScript provides interfaces to work with the HTTP protocol but in a slightly different way. The JavaScript HTTP interfaces are designed in such a way that it supports **asynchronous communication**. This means that when an HTTP request is sent using these

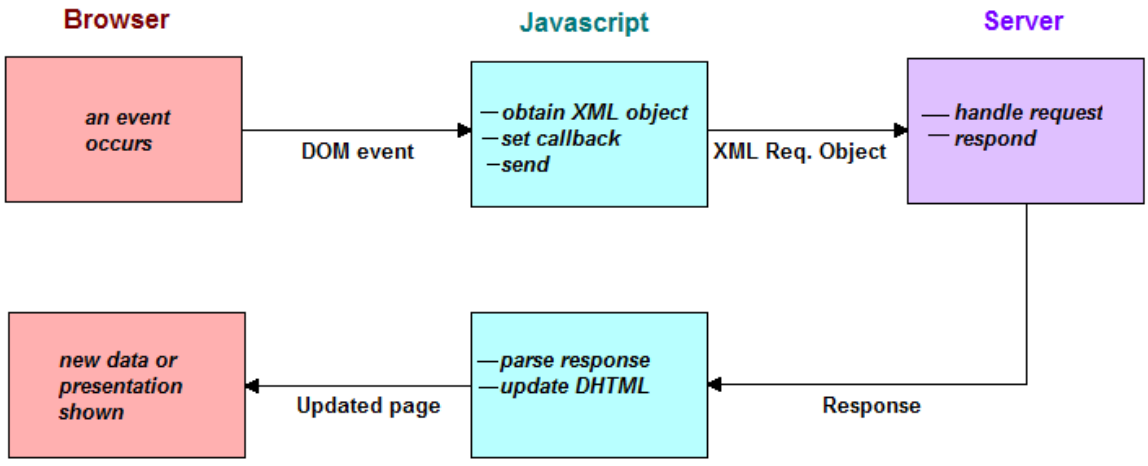
interfaces to the web server, the sender does not get blocked. It continues its normal execution. This is the **"asynchronous"** part of the AJAX. Whenever response comes back, the sender is interrupted. If the sender is properly configured, a JavaScript function is executed as part of this interrupt. The data returned by the server are processed in this function and a specific part of the web page is modified.



Processing steps to implement AJAX- based applications

To write the AJAX- based applications, the programmer should follow the following steps:

1. **Define the User Input and Output**
2. **Create an XMLHttpRequest Object in JavaScript**
3. **Specify the Handler for tracking the request object**
4. **Opening HTTP connection using XMLHttpRequest Object**
5. **Sending the request to the web server.**
6. **Receiving the response and shows the response text to the user.**



Step 1 : Define the User Input and Output

The web application must consists of the component that allows the user to input data and then to see the results. This can be achieved in HTML by using:

- A simple form for the user input
- Div or span areas for displaying any outputs

For example

```
<form name=user_input>
Enter Roll Number :<br>
<input type=”text”  name=”rollno” size 10 />
<input type=”button” value=”GetResult” onClick=”send_request()” />
</form>
<span id=”result”></span>
</body>
</html>
```

Step 2 : Create an XMLHttpRequest Object in JavaScript

The heart of **AJAX** technology is the **XMLHttpRequest** object. It provides a set of useful properties and methods that are used to send HTTP request to and retrieve data from the web server.

So, the programmer must first create an instance of the XMLHttpRequest object. The procedure of creating this object is different for different browsers.

There are 3 ways to create this object:

- For most of the browsers:
  - **var request = new XMLHttpRequest();**
- For some IE versions:
  - **var request = new ActiveXObject("Microsoft.XMLHTTP");**
- For other versions of IE:
  - **var request = new ActiveXObject("Msxml2.XMLHTTP");**

For Example

```
try
{
        request= new XMLHttpRequest( );
}
catch( e1 )
{
    alert(“AJAX is not supported”);
}
```

Step 3: Specify the Handler for tracking the request object

When we use an asynchronous request, we cannot be sure when the response will come back. So, before sending the HTTP request to the web server, we must specify the JavaScript function to be called when the HTTP response from the server comes back. We call this function as **"Response Handler"** or **"Callback function".** The response handler is specified using the **"onreadystatechange"** property of the xmlhttp object. As the name implies, every time the ready state of the request changes, the handler function gets executed.

For example

```
function handler( )
{
  // code to extract and use the data sent by the server.
}

// specify the function handler () as the response handler
request.onreadystatechange=handler;
```

The handler may also be an anonymous function as follows

```
request.onreadystatechange= function ( )
                            {
                                    // code to extract and use the data sent by the server.
                            }
```

Step 4 : Opening HTTP connection using XMLHttpRequest Object

After the creation of XMLHttpRequest object, server request should be prepared with the **open( )** method.

Syntax :

request.open(method, URL, async)

- "method" can be '**GET**' or '**POST**'
- "URL" - determines where the request will be sent
  - If GET => parameters will be added to the end of the URL
  - If POST => parameters will be added in the package
- "async" (**true/false**), specifies whether the request should be handled asynchronously or not – true is default

Step 5 : Sending the request to the web server

After the creation of XMLHttpRequest object, request should be sent to the server with **send()** method.

Syntax:

- **request.send(null);**
  - Used if the method is GET in open( )
- **request.send([data]);**
  - Used if the method is POST in open( )

If the **POST** method is used in open( ), the parameters are included in the body of the HTTP request message. In that case, we have to first set the **Content-type** header to **"application/x-www-form-urlencoded"**. For example

```
request.open("POST", "getresult.php", true);
request.setRequestHeader("Content-type","application/x-www-form-urlencoded")
request.send("rollno=1001");
```

Step 6 : Receiving the response and shows the response text to the user

The process of communicating with the server, from sending an HTTP request to getting a response involves several states. The state can be one of the following

State	Description
0	The request is not initialized
1	The request has been set up
2	The request has been sent
3	The request is in process
4	The request is complete

Every time the state changes, the specified handler function get executed. But we are particularly interested in the state "**Completed**" because the response data is available only in that ready state. Before doing anything in the handler function, make sure that the state is "completed" by verifying the "**readyState**" property of the request object.

The response from the server can be retrieved from either **responseText** property or **responseXML** property of XMLHttpRequest object.

For example

```
function handler( )
{
        if(request.readyState == 4)
        {
                var sp=document.getElelemntById("result");
                sp.innterHTML="<b>"+request.responseText+"</>";
        }
}
```

Example AJAX Program

SimpleAjax.html

```
<html>
  <head>
    <script type='text/javascript' src='simpleajax.js'></script>
    <title>PHP AJAX Example</title>
  </head>
  <body>
    <input type='button' onclick='MakeRequest();' value='Use AJAX!!!!'/>
    <div id='ResponseDiv'>
      This is a div to hold the response.
    </div>
  </body>
</html>
```

SimpleAjax.js

```
function MakeRequest()
{
  var xmlHttp;

  try
  {

    xmlHttp = new XMLHttpRequest();
  }
  catch(e)
  {

       alert("Your browser does not support AJAX!")
  }
   xmlHttp.onreadystatechange = function()
                                {
                                   if(xmlHttp.readyState == 4)
                                   {
                                          HandleResponse(xmlHttp.responseText);
                                   }
                                }

  xmlHttp.open("GET", "SimpleAjax.php", true);
  xmlHttp.send(null);
}

function HandleResponse(response)
{
  document.getElementById('ResponseDiv').innerHTML = response;
}
```

SimpleAjax.php

```
<?php
echo "This is a php response to your request!!!!!!";
?>
```