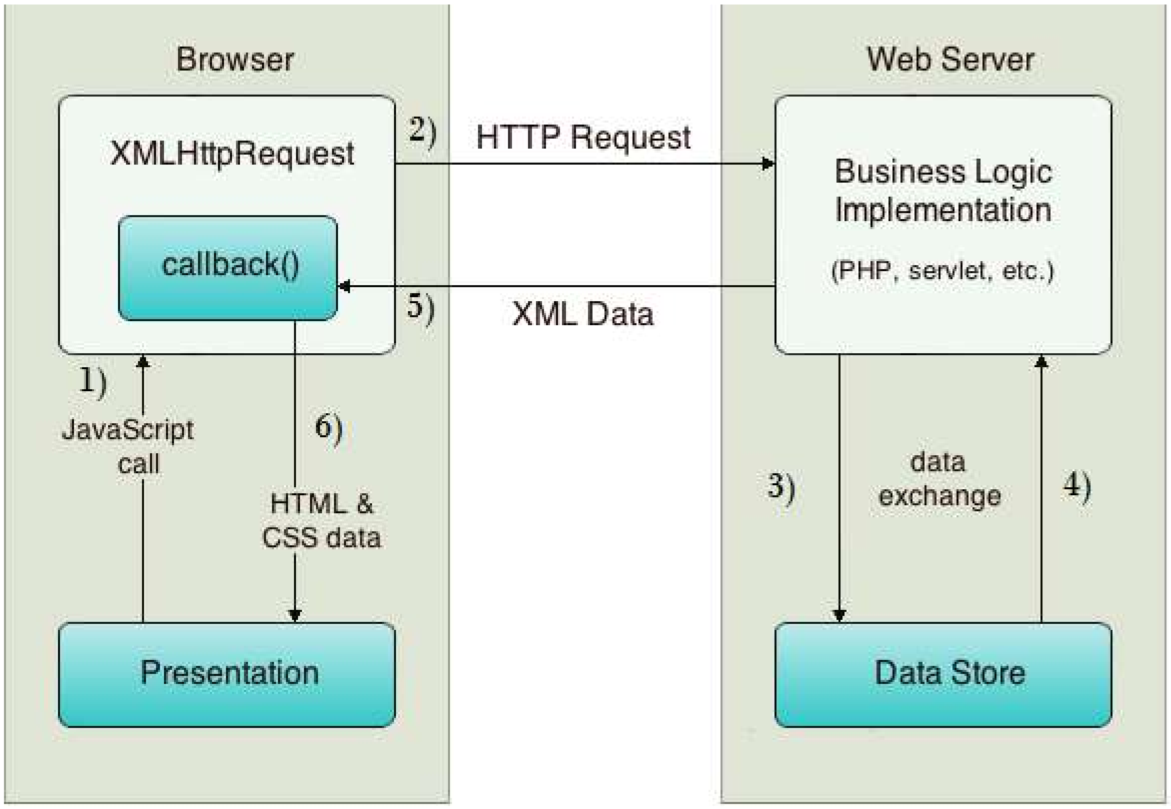
Ajax

How AJAX works?

AJAX communicates with the server using XMLHttpRequest object. Let's try to understand the flow of ajax or how ajax works by the image displayed below.



As you can see in the above example, XMLHttpRequest object plays a important role.

1. User sends a request from the UI and a javascript call goes to XMLHttpRequest object.
2. HTTP Request is sent to the server by XMLHttpRequest object.
3. Server interacts with the database using JSP, PHP, Servlet, ASP.net etc.
4. Data is retrieved.
5. Server sends XML data or JSON data to the XMLHttpRequest callback function.
6. HTML and CSS data is displayed on the browser.

Understanding XMLHttpRequest

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.

Properties of XMLHttpRequest object

roperty	Description
onReadyStateChange	It is called whenever readystate attribute changes. It must not be used with synchronous requests.
readyState	Represents the state of the request. It ranges from 0 to 4. **0** UNOPENED open() is not called. **1** OPENED open is called but send() is not called. **2** HEADERS_RECEIVED send() is called, and headers and status are available. **3** LOADING Downloading data; responseText holds the data. **4** DONE The operation is completed fully.
reponseText	returns response as text.
responseXML	returns response as XML

Methods of XMLHttpRequest object

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.

There are four ready states in Ajax:

- Initialization
- Request
- Process
- Ready

A new object of **XMLHttpRequest** is created like this:

```
//Creating a new XMLHttpRequest object
var xmlhttp;
if (window.XMLHttpRequest)
{
      xmlhttp = new XMLHttpRequest(); //for IE7+,Firefox,Chrome,Opera,Safari
}
else
{
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP"); //for IE6, IE5
}
```

open (method, url, isAsync, userName, password)

```
xmlhttp.open("GET","report_data.xml",true);
xmlhttp.open("GET","sensitive_data.xml",false);
xmlhttp.open("POST","saveData",true,"myUserName","somePassord");
```

setRequestHeader(name, value)

```
/Tells server that this call is made for ajax purposes.
xmlhttp.setRequestHeader('X-Requested-With', 'XMLHttpRequest');
```

send(payload)

```
xmlhttp.send(null); //Request with no data in request body; Mostly used in GET
requests.
xmlhttp.send( {"id":"23423"} ); //Request with data in request body; Mostly used
in POST/ PUT requests.
```

Synchronous and Asynchronous requests

```
xmlhttp.open("GET", "report_data.xml", true); //Asynchrnonos request as third
parameter is true
xmlhttp.open("GET", "report_data.xml", false); Synchrnonos request as third
parameter is false
```

Example synchronous request

```
var request = new XMLHttpRequest();
request.open('GET', '/bar/foo.txt', false);  //"false" makes the request
synchronous
request.send(null);

if(request.status === 200)
{
    //request successful; handle the response
}
else
{
    //Request failed; Show error message
}
```

Example asynchronous request

```
var request = new XMLHttpRequest();
request.open('GET', '/bar/foo.txt', true);  //"true" makes the request
asynchronous

request.onreadystatechange = function() {
    if (request.readyState == 4) {
        if (request.status == 200)
        {
            //request succeed
        }
        else
        {
            //request failed
        }
    }
};

request.send(null);
```

Ready state value:

0 : request not initialized
1 : server connection established
2 : request received
3 : processing request
4 : request finished and response is ready to be handled

Handling returned response from server

To get the response from a server, responseText or responseXML property of the XMLHttpRequest object is used. If the response from the server is XML, and you want to parse it as an XML object, use the responseXML property. If the response from the server is not XML, use the responseText property.

responseText : Get the response from server as a string
responseXML : Get the response from server as XML

Example:

```
if (xmlhttp.readyState == 4) {
    if (xmlhttp.status == 200)
    {
        document.getElementById("message").innerHTML = xmlhttp.responseText;
    }
    else {
        alert('Something is wrong !!');
    }
}
```

Demo application code

For demonstration purpose, I am creating a very simple hello world application. In this application, webpage sends a ajax GET request to query the current server's system time. In response, server sends the current time. Easy enough.

Asynchronous request example

To enable the webpage to send such request, I have written following javascript code in JSP page:

```
function ajaxAsyncRequest(reqURL)
{
    //Creating a new XMLHttpRequest object
    var xmlhttp;
    if (window.XMLHttpRequest){
        xmlhttp = new XMLHttpRequest(); //for IE7+, Firefox, Chrome, Opera,
Safari
    } else {
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP"); //for IE6, IE5
    }
    //Create a asynchronous GET request
    xmlhttp.open("GET", reqURL, true);

    //When readyState is 4 then get the server output
    xmlhttp.onreadystatechange = function() {
        if (xmlhttp.readyState == 4) {
            if (xmlhttp.status == 200)
            {
                document.getElementById("message").innerHTML =
xmlhttp.responseText;
                //alert(xmlhttp.responseText);
            }
            else
            {
                alert('Something is wrong !!');
            }
        }
    };

    xmlhttp.send(null);
}
```

and to fire the ajax request, a button should be clicked which is written as:

```
<input type="button" value="Show Server
Time" onclick='ajaxAsyncRequest("get-current-time")'/>
```

To handle the request on server side, I have written a servlet like this (here could be any java servlet class:

```
public class GetTimeServlet extends HttpServlet
{
    private static final long serialVersionUID = 1L;

    public void doGet (HttpServletRequest request,HttpServletResponse response)
            throws ServletException, IOException
    {
        response.setHeader("Cache-Control", "no-cache");
        response.setHeader("Pragma", "no-cache");
        PrintWriter out = response.getWriter();
        Date currentTime= new Date();
        String message = String.format("Currently time is %tr on
%tD.",currentTime, currentTime);
        out.print(message);
    }
}
```

JQuery

JQuery is probably the post popular among its alternatives. It has got its own developer community which is highly active also. A sample code for sending ajax request using jquery will be like is:

```
/*clear result div*/
   $("#result").html('');

  /* get some values from elements on the page: */
   var values = $(this).serialize();

  /* Send the data using post and put the results in a div */
  request =$.ajax({
      url: "ajaxRequest",
      type: "post",
      data: values,
      success: function(){
           $("#result").html('submitted successfully');
      },
      error:function(){
          $("#result").html('there is error while submit');
      }
  });
```

Thank You!

Anil Kumar