What is Ajax?

Asynchronous JavaScript and XML or Ajax for short is new web development technique used for the development of most interactive website. Ajax helps you in making your web application more interactive by retrieving small amount of data from web server and then showing it on your application. You can do all these things without refreshing your page.

Usually in all the web applications, the user enters the data into the form and then clicks on the submit button to submit the request to the server. Server processes the request and returns the view in new page (by reloading the whole page). This process is inefficient, time consuming, and a little frustrating for you user if the only the small amount of data exchange is required. For example in a user registration form, this can be frustrating thing for the user, as whole page is reloaded only to check the availability of the user name. Ajax will help in making your application more interactive. With the help of Ajax you can tune your application to check the availability of the user name without refreshing the whole page.

Understanding the technology behind Ajax

Ajax is not a single technology, but it is a combination of many technologies. These technologies are supported by modern web browsers. Following are techniques used in the Ajax applications.

- **JavaScript:**
  JavaScript is used to make a request to the web server. Once the response is returned by the webserver, more JavaScript can be used to update the current page. DHTML and CSS is used to show the output to the user. JavaScript is used very heavily to provide the dynamic behavior to the application.
- **Asynchronous Call to the Server:**
  Most of the Ajax application used the XMLHttpRequest object to send the request to the web server. These calls are Asynchronous and there is no need to wait for the response to come back. User can do the normal work without any problem.
- **XML:**
  XML may be used to receive the data returned from the web server. JavaScript can be used to process the XML data returned from the web server easily.

How Ajax Works?

When user first visits the page, the Ajax engine is initialized and loaded. From that point of time user interacts with Ajax engine to interact with the web server. The Ajax engine operates asynchronously while sending the request to the server and receiving the response from server. Ajax life cycle within the web browser can be divided into following stages:

- **User Visit to the page:** User visits the URL by typing URL in browser or clicking a link from some other page.
- **Initialization of Ajax engine:**
  When the page is initially loaded, the Ajax engine is also initialized. The Ajax engine can also be set to continuously refresh the page content without refreshing the whole page.

- **Event Processing Loop:**
  Browser event may instruct the Ajax engine to send request to server and receive the response data
  Server response - Ajax engine receives the response from the server. Then it calls the JavaScript call back functions
  Browser (View) update - JavaScript request call back functions is used to update the browser. DHTML and css is used to update the browser display.

Benefits of Ajax

Ajax is new very promising technology, which has become extremely popular these days. Here are the benefits of using Ajax:

- Ajax can be used for creating rich, web-based applications that look and works like a desktop application
- Ajax is easy to learn. Ajax is based on JavaScript and existing technologies like XML, CSS, DHTML etc. So, its very easy to learn Ajax.
- Ajax can be used to develop web applications that can update the page data continuously without refreshing the whole page.
- **XMLHttpRequest** - It is used for making requests to the non-Ajax pages. It supports all kind of HTTP request type.
- **IFrame** - It can make requests using both POST and GET methods. It supports every modern browser. It supports asynchronous file uploads.
- **Cookies** - In spite of implementation difference among browsers it supports large number of browsers
- **The interface** is much **responsive**, instead of the whole page; a section of the page is transferred at a time.
- **Waiting time is reduced**
- **Traffic** to and from the server is **reduced.**

Disadvantages:

1. **XMLHttpRequest** -In the older version of IE (5 & 6) ActiveX to be enabled, this feature is available in new browsers and latest version of few.
2. **IFrame** - Requests have to be asynchronous. The design of the Server pages must be compatible with IFrame requests. There is an implementation difference among different browsers. Unnecessary history records can be left on the history record of the browser. Request size is increased due to the fact that data is URL-encoded.
3. **Cookies** - It prohibits no synchronous requests, it does not cope with large requests/results The server pages requires to be designed to work with cookie requests.

XMLHttpRequest Object:

As the use of XML and web services is increasing day by day, it is good to connect an HTML page with XML, with the help of that we can get interim updates so easily without reloading the page. It is possible due to the XMLHttpRequest object, which is responsible for retrieving and submitting the XML data directly from the client side. It

relies on Document Object Model (DOM) to convert the retrieved XML data into HTML.

Microsoft first implemented the XMLHttpRequest object in IE 5 for Windows as an ActiveX object. Afterward Mozilla and Apple's Safari also implemented in their own style.

XMLHttpRequest object facilitates the web developer a lot because with the help of this object you can update the page with new data without reloading the page, communicate with server in the background, request and receive data from the web server.

Creating an XMLHttpRequest object.

We can create an instance of the XMLHttpRequest in most of the modern popular browsers, and in the old versions of the browsers we need to create an object of ActiveXObject.

```
var xmlobj=new XMLHttpRequest ();
var activeobj=new ActiveXObject("Microsoft.XMLHTTP");
```

How to use an XMLHttpRequest ?

Important Methods

The XMLHttpRequest object has 2 important methods:

- The open() method
- The send() method

Sending an AJAX Request to a Server

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

The open() method takes three arguments. The first argument defines which method to use (GET or POST). The second argument specifies the name of the server resource (URL). The third argument specifies if the request should be handled asynchronously.

The send() method sends the request off to the server. If we assume that requested a file called "time.asp", the code would be:

```
url="time.asp"
xmlhttp.open("GET",url,true);
xmlhttp.send(null);
```

In the example we assume that the current web page and the requested resource are both in the same file directory.

Important Properties

The XMLHttpRequest object has 3 important properties:

- The responseText property
- The readyState property
- The onreadystatechange property

The responseText property

The XMLHttpRequest object stores any data retrieved from a server as a result of a server request in its **responseText property**.

In the previous chapter we copied the content of the responsText property into our HTML with the following statement:

```
document.getElementById('test').innerHTML=xmlhttp.responseText
```

XMLHttpRequest Open - Using False

In the examples (from the previous pages) we used this simplified syntax:

```
xmlhttp.open("GET",url,false);
xmlhttp.send(null);
document.getElementById('test').innerHTML=xmlhttp.responseText;
```

The third parameter in the open call is "false". This tells the XMLHttpRequest object to wait until the server request is completed before next statement is executed.

For small applications and simple server request, this might be OK. But if the request takes a long time or cannot be served, this might cause your web application to hang or stop.

XMLHttpRequest Open - Using True

By changing the third parameter in the open call to "true", you tell the XMLHttpRequest object to continue the execution after the request to the server has been sent.

Because you cannot simply start using the response from the server request before you are sure the request has been completed, you need to set the **onreadychange property** of the XMLHttpRequest, to a function (or name of a function) to be executed after completion.

In this onreadystatechange function you must test the **readyState property** before you can use the result of the server call.

Simply change the code to:

```
xmlhttp.onreadystatechange=function()
{
if(xmlhttp.readyState==4)
  {document.getElementById('test').innerHTML=xmlhttp.responseText}
}
xmlhttp.open("GET",url,true);
xmlhttp.send(null);
```

The readyState property

The readyState property holds the status of the server's response.

Possible values for the readyState property:

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

The onreadystatechange property

The onreadystatechange property stores a function (or the name of a function) to be called automatically each time the readyState property changes.

You can store a full function in the property like this:

```
xmlhttp.onreadystatechange=function()
{
if(xmlhttp.readyState==4)
  {document.getElementById('test').innerHTML=xmlhttp.responseText}
}
xmlhttp.open("GET",url,true);
xmlhttp.send(null);
```

Or you can store the name of a function like this:

```
xmlhttp.onreadystatechange=state_Change()
xmlhttp.open("GET",url,true);
xmlhttp.send(null);
...
...
...
function state_Change()
{
if(xmlhttp.readyState==4)
  {document.getElementById('test').innerHTML=xmlhttp.responseText}
}
```

Example:

Please consult lecture server on CMS.