UNIT-5

AJAX A NEW APPROACH

AJAX Introduction

AJAX is the art of exchanging data with a server, and updating parts of a web page - without reloading the whole page.

AJAX = Asynchronous JavaScript and XML.

In short; AJAX is about loading data in the background and display it on the webpage, without reloading the whole page.

Examples of applications using AJAX: Gmail, Google Maps, Youtube, and Facebook tabs.

AJAX is a web development technique for creating interactive web applications. If you know JavaScript, HTML, CSS, and XML, then you need to spend just one hour to start with AJAX.

Why to Learn Ajax?

AJAX stands for **A**synchronous **Ja**vaScript and **X**ML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

- Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and 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.

Rich Internet Application Technology

AJAX is the most viable Rich Internet Application (RIA) technology so far. It is getting tremendous industry momentum and several tool kit and frameworks are emerging. But at the same time, AJAX has browser incompatibility and it is supported by JavaScript, which is hard to maintain and debug.

AJAX is Based on Open Standards

AJAX is based on the following open standards −

- Browser-based presentation using HTML and Cascading Style Sheets (CSS).
- Data is stored in XML format and fetched from the server.
- Behind-the-scenes data fetches using XMLHttpRequest objects in the browser.
- JavaScript to make everything happen.

- **Creating XMLHttpRequest Object:**

AJAX – XMLHttpRequest

The XMLHttpRequest object is the key to AJAX. It has been available ever since Internet Explorer 5.5 was released in July 2000, but was not fully discovered until AJAX and Web 2.0 in 2005 became popular.

XMLHttpRequest (XHR) is an API that can be used by JavaScript, JScript, VBScript, and other web browser scripting languages to transfer and manipulate XML data to and from a webserver using HTTP, establishing an independent connection channel between a webpage's Client-Side and Server-Side.

The data returned from XMLHttpRequest calls will often be provided by back-end databases. Besides XML, XMLHttpRequest can be used to fetch data in other formats, e.g. JSON or even plain text.

You already have seen a couple of examples on how to create an XMLHttpRequest object.

Listed below are some of the methods and properties that you have to get familiar with.

XMLHttpRequest Methods

- **abort()**

  Cancels the current request.

- **getAllResponseHeaders()**

  Returns the complete set of HTTP headers as a string.

- **getResponseHeader(headerName)**

  Returns the value of the specified HTTP header.

- **open(method, URL)**
- **open(method, URL, async)**
- **open(method, URL, async, userName)**
- **open(method, URL, async, userName, password)**

  Specifies the method, URL, and other optional attributes of a request.

  The method parameter can have a value of "GET", "POST", or "HEAD". Other HTTP methods such as "PUT" and "DELETE" (primarily used in REST applications) may be possible.

  The "async" parameter specifies whether the request should be handled asynchronously or not. "true" means that the script processing carries on after the send() method without waiting for a response, and "false" means that the script waits for a response before continuing script processing.

- **send(content)**

  Sends the request.

- **setRequestHeader(label, value)**

  Adds a label/value pair to the HTTP header to be sent.

XMLHttpRequest Properties

- **onreadystatechange**

  An event handler for an event that fires at every state change.

- **readyState**

  The readyState property defines the current state of the XMLHttpRequest object.

  The following table provides a list of the 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 completed.

readyState = 0 After you have created the XMLHttpRequest object, but before you have called the open() method.

readyState = 1 After you have called the open() method, but before you have called send().

readyState = 2 After you have called send().

readyState = 3 After the browser has established a communication with the server, but before the server has completed the response.

readyState = 4 After the request has been completed, and the response data has been completely received from the server.

- **responseText**

  Returns the response as a string.

- **responseXML**

  Returns the response as XML. This property returns an XML document object, which can be examined and parsed using the W3C DOM node tree methods and properties.

- **status**

  Returns the status as a number (e.g., 404 for "Not Found" and 200 for "OK").

- **statusText**

Returns the status as a string (e.g., "Not Found" or "OK").

- **Integrating AJAX with PHP**

AJAX and PHP

AJAX PHP Example

The following example will demonstrate how a web page can communicate with a web server while a user type characters in an input field:

Example

Start typing a name in the input field below:

First name:

Suggestions:

Example Explained

In the example above, when a user types a character in the input field, a function called "showHint()" is executed.

The function is triggered by the onkeyup event.

Here is the HTML code:

Example

```
<html>
<head>
<script>
function showHint(str) {
    if (str.length == 0) {
        document.getElementById("txtHint").innerHTML = "";
        return;
    } else {
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
```

```
            document.getElementById("txtHint").innerHTML = this.responseText;
        }
      };
      xmlhttp.open("GET", "gethint.php?q=" + str, true);
      xmlhttp.send();
  }
}
</script>
</head>
<body>

<p><b>Start typing a name in the input field below:</b></p>
<form>
First name: <input type="text" onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>
</body>
</html>
```

Code explanation:

First, check if the input field is empty (str.length == 0). If it is, clear the content of the txtHint placeholder and exit the function.

However, if the input field is not empty, do the following:

- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a PHP file (gethint.php) on the server
- Notice that q parameter is added to the url (gethint.php?q="+str)
- And the str variable holds the content of the input field

The PHP File - "gethint.php"

The PHP file checks an array of names, and returns the corresponding name(s) to the browser:

```
<?php
// Array with names
$a[] = "Anna";
$a[] = "Brittany";
$a[] = "Cinderella";
```

```php
$a[] = "Diana";
$a[] = "Eva";
$a[] = "Fiona";
$a[] = "Gunda";
$a[] = "Hege";
$a[] = "Inga";
$a[] = "Johanna";
$a[] = "Kitty";
$a[] = "Linda";
$a[] = "Nina";
$a[] = "Ophelia";
$a[] = "Petunia";
$a[] = "Amanda";
$a[] = "Raquel";
$a[] = "Cindy";
$a[] = "Doris";
$a[] = "Eve";
$a[] = "Evita";
$a[] = "Sunniva";
$a[] = "Tove";
$a[] = "Unni";
$a[] = "Violet";
$a[] = "Liza";
$a[] = "Elizabeth";
$a[] = "Ellen";
$a[] = "Wenche";
$a[] = "Vicky";

// get the q parameter from URL
$q = $_REQUEST["q"];

$hint = "";

// lookup all hints from array if $q is different from ""
if ($q !== "") {
    $q = strtolower($q);
    $len=strlen($q);
    foreach($a as $name) {
        if (stristr($q, substr($name, 0, $len))) {
            if ($hint === "") {
```

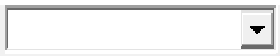
```
                $hint = $name;
            } else {
                $hint .= ", $name";
            }
        }
    }
}

// Output "no suggestion" if no hint was found or output correct values
echo $hint === "" ? "no suggestion" : $hint;
?>
```

- **Retrieving data from a database using PHP and AJAX**

AJAX Database Example

The following example will demonstrate how a web page can fetch information from a database with AJAX:

Example



Person info will be listed here...

Example Explained - The MySQL Database

The database table we use in the example above looks like this:

id	FirstName	LastName	Age	Hometown	Job
1	Peter	Griffin	41	Quahog	Brewery
2	Lois	Griffin	40	Newport	Piano Teacher

3	Joseph	Swanson	39	Quahog	Police Officer
4	Glenn	Quagmire	41	Quahog	Pilot

Example Explained

In the example above, when a user selects a person in the dropdown list above, a function called "showUser()" is executed.

The function is triggered by the onchange event.

Here is the HTML code:

Example

```
<html>
<head>
<script>
function showUser(str) {
    if (str == "") {
        document.getElementById("txtHint").innerHTML = "";
        return;
    } else {
        if (window.XMLHttpRequest) {
            // code for IE7+, Firefox, Chrome, Opera, Safari
            xmlhttp = new XMLHttpRequest();
        } else {
            // code for IE6, IE5
            xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
        }
        xmlhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("txtHint").innerHTML = this.responseText;
            }
        };
        xmlhttp.open("GET","getuser.php?q="+str,true);
        xmlhttp.send();
    }
```

```
}
</script>
</head>
<body>

<form>
<select name="users" onchange="showUser(this.value)">
  <option value="">Select a person:</option>
  <option value="1">Peter Griffin</option>
  <option value="2">Lois Griffin</option>
  <option value="3">Joseph Swanson</option>
  <option value="4">Glenn Quagmire</option>
  </select>
</form>
<br>
<div id="txtHint"><b>Person info will be listed here...</b></div>

</body>
</html>
```

Code explanation:

First, check if person is selected. If no person is selected (str == ""), clear the content of txtHint and exit the function. If a person is selected, do the following:

- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a file on the server

The PHP File

The page on the server called by the JavaScript above is a PHP file called "getuser.php".

The source code in "getuser.php" runs a query against a MySQL database, and returns the result in an HTML table:

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
```

```
    width: 100%;
    border-collapse: collapse;
}

table, td, th {
    border: 1px solid black;
    padding: 5px;
}

th {text-align: left;}
</style>
</head>
<body>

<?php
$q = intval($_GET['q']);

$con = mysqli_connect('localhost','peter','abc123','my_db');
if (!$con) {
    die('Could not connect: ' . mysqli_error($con));
}

mysqli_select_db($con,"ajax_demo");
$sql="SELECT * FROM user WHERE id = '".$q."'";
$result = mysqli_query($con,$sql);

echo "<table>
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Age</th>
<th>Hometown</th>
<th>Job</th>
</tr>";
while($row = mysqli_fetch_array($result)) {
    echo "<tr>";
    echo "<td>" . $row['FirstName'] . "</td>";
    echo "<td>" . $row['LastName'] . "</td>";
    echo "<td>" . $row['Age'] . "</td>";
    echo "<td>" . $row['Hometown'] . "</td>";
```

```
    echo "<td>" . $row['Job'] . "</td>";
    echo "</tr>";
}
echo "</table>";
mysqli_close($con);
?>
</body>
</html>
```

Explanation: When the query is sent from the JavaScript to the PHP file, the following happens:

1. PHP opens a connection to a MySQL server
2. The correct person is found
3. An HTML table is created, filled with data, and sent back to the "txtHint" placeholder

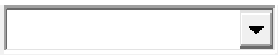
➢ Handling Xml Data Using PHP and AJAX:

AJAX can be used for interactive communication with an XML file.

AJAX XML Example

The following example will demonstrate how a web page can fetch information from an XML file with AJAX:

Example



TITLE: The very best of
ARTIST: Cat Stevens
COUNTRY: UK
COMPANY: Island
PRICE: 8.90
YEAR: 1990

Example Explained - The HTML Page

When a user selects a CD in the dropdown list above, a function called "showCD()" is executed. The function is triggered by the "onchange" event:

```
<html>
<head>
<script>
function showCD(str) {
  if (str=="") {
    document.getElementById("txtHint").innerHTML="";
    return;
  }
  if (window.XMLHttpRequest) {
    // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
  } else {  // code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
  xmlhttp.onreadystatechange=function() {
    if (this.readyState==4 && this.status==200) {
      document.getElementById("txtHint").innerHTML=this.responseText;
    }
  }
  xmlhttp.open("GET","getcd.php?q="+str,true);
  xmlhttp.send();
}
</script>
</head>
<body>

<form>
Select a CD:
<select name="cds" onchange="showCD(this.value)">
<option value="">Select a CD:</option>
<option value="Bob Dylan">Bob Dylan</option>
<option value="Bee Gees">Bee Gees</option>
<option value="Cat Stevens">Cat Stevens</option>
</select>
</form>
```

```
<div id="txtHint"><b>CD info will be listed here...</b></div>

</body>
</html>
```

The showCD() function does the following:

- Check if a CD is selected
- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a file on the server
- Notice that a parameter (q) is added to the URL (with the content of the dropdown list)

The PHP File

The page on the server called by the JavaScript above is a PHP file called "getcd.php".

The PHP script loads an XML document, "cd_catalog.xml", runs a query against the XML file, and returns the result as HTML:

```
<?php
$q=$_GET["q"];

$xmlDoc = new DOMDocument();
$xmlDoc->load("cd_catalog.xml");

$x=$xmlDoc->getElementsByTagName('ARTIST');

for ($i=0; $i<=$x->length-1; $i++) {
  //Process only element nodes
  if ($x->item($i)->nodeType==1) {
    if ($x->item($i)->childNodes->item(0)->nodeValue == $q) {
      $y=($x->item($i)->parentNode);
    }
  }
}

$cd=($y->childNodes);

for ($i=0;$i<$cd->length;$i++) {
```

```
  //Process only element nodes
  if ($cd->item($i)->nodeType==1) {
    echo("<b>" . $cd->item($i)->nodeName . ":</b> ");
    echo($cd->item($i)->childNodes->item(0)->nodeValue);
    echo("<br>");
  }
}
?>
```

When the CD query is sent from the JavaScript to the PHP page, the following happens:

1. PHP creates an XML DOM object
2. Find all <artist> elements that matches the name sent from the JavaScript
3. Output the album information (send to the "txtHint" placeholder)