AJAX

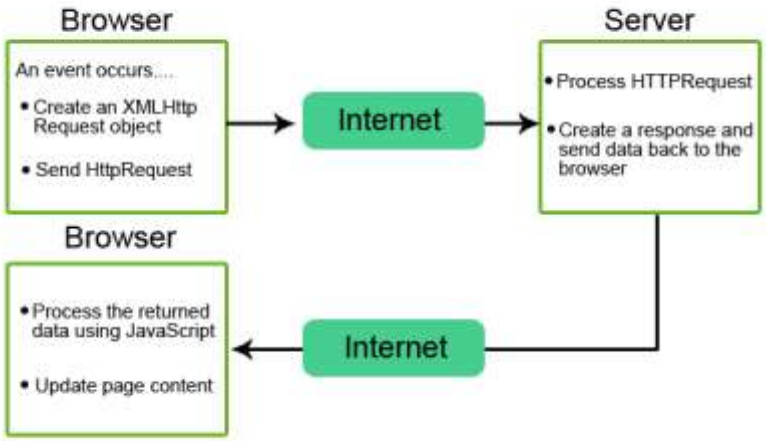
- AJAX stands for Asynchronous JavaScript and XML.
- AJAX is a technique for creating better, faster and more interactive web applications with the help of XML, HTML, CSS and Java Script.
- AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.
- Classic web pages (which do not use AJAX) must reload the entire page if the content should change.

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

- AJAX is based on internet standards and uses a combination of:
  - Browser built-in XMLHttpRequest object - To request or exchange data asynchronously from a web server
  - JavaScript/DOM - To display/interact with the information
  - CSS - To style the data
  - XML - often used as the format for transferring data

- AJAX applications are browser and platform independent.

How does AJAX Works?



1. An event occurs in a web page (the page is loaded or a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

XMLHttpRequest object:

The XMLHttpRequest object can be used to exchange data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

XMLHttpRequest Object Methods:

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(*method, url, async, user, psw*)	Specifies the request *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password
send()	Sends the request to the server Used for GET requests
send(*string*)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

XMLHttpRequest Object Properties:

Property	Description
onload	Defines a function to be called when the request is recieved (loaded)
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found"
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Basic Example:

To create any AJAX application follows below steps:

1. **Create an XMLHttpRequest object** – eg. variable = new XMLHttpRequest();
2. **Define a callback function** - A callback function is a function passed as a parameter to another function. In this case, the callback function should contain the code to execute when the response is ready.
   Eg. xhttp.onload = function() {
   // What to do when the response is ready
   }
   Eg. xhttp.onreadystatechange = function() {
   if (this.readyState == 4 && this.status == 200) {
   // What to do when the response is ready
   } };
3. **Open the XMLHttpRequest object** – Specify the request
4. **Send a Request to a server** - To send a request to a server, you can use the open() and send() methods of the XMLHttpRequest object:
   Eg. xhttp.open("GET", "ajax_info.txt");
   xhttp.send();

Where to Use It?

Login forms: Eg: user can type their login credentials in the original page form; their software will send a request to the server to logged in and the page will be updated as needed.

Auto-complete: When query is running in the Google search bar with the help of auto fill settings, suggestions will be shown in below drop down.

Rating and Voting: The voting can decide the site's main content in web pages like live voting poll.

Updating with user content: When a user posts a tweet, it will be added to their feed and everything is updated.

Now it is used by tweeter to run their trending topics page

- Form submission and validation
- It makes web applications quicker and the numbers of responses are also reduced.
- Light-boxes are used nowadays instead of pop-ups
- flash application

Why Ajax is Used?

Ajax is a web developer's long term dream because the user can do the following things:

1. Without reloading the page, the user request can be updated

2. After the page is loaded, it generates data from the server.

3. Receive data from the server after the page has loaded.

4. In the background, sends data to the server

Examples of Ajax Application:

Google suggests that auto-complete options will be offered while typing when a user enters the search query in the Google search bar. Suggestions given by Google can be navigated by using operational keys.

Yahoo maps are easier while operating and user experiences more fun. This Map uses Ajax to drag the entire map with the mouse without using buttons that will be at ease to the user.

Google maps are general applications that use Ajax. This is a real-time application in which the user can manipulate the data and change the view settings. Ajax directly works on a web browser without any plugin installations. Firstly, only Microsoft internet explorer used Ajax, but due to its reliability, more web applications like chrome, Mozilla… Etc. using this.

Ajax applications use an intermediate engine that acts as a bridge between browser and server. Ajax is not a programming language.

Example-1: how a web page can communicate with a web server while a user type characters in an input field.

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

Example1.html

```
<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 action="">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname" onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>
</body>
</html>
```

Gethint.php

```
<?php
// Array with names
$a[] = "Anna";
$a[] = "Brittany";
$a[] = "Cinderella";
$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 $name) {
   // The stristr() function searches for the first occurrence of a string inside another string.
  // The substr() function returns a part of a string.
    if (stristr($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;
?>
```

Example-2: how a web page can fetch information from a database with AJAX.
Note: Assume you have database as per user.

Code explanation:
First, check if person is selected or not. 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
- Notice that a parameter (q) is added to the URL (with the content of the dropdown list)

Example2.html

```
<html>
<head>
<script>
function showUser(str) {
  if (str == "") {
    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","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">Mr. XYZ</option>
  <option value="2">Mr.ABCD</option>
```

```
  <option value="3">Mr.PQR</option>
  <option value="4">Karan Patel</option>
  </select>
</form>
<br>
<div id="txtHint"><b>Person info will be listed here...</b></div>

</body>
</html>
```

Getuser.php

```
<!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']);

$con = mysqli_connect('localhost','root',' ','wt_db1');
if (!$con) {
  die('Could not connect: ' . mysqli_error($con));
}

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

echo "<table>
<tr>
```

```
<th>Id</th>
<th>Name</th>
<th>Salary</th>
<th>Job</th>
</tr>";
while($row = mysqli_fetch_array($result)) {
  echo "<tr>";
  echo "<td>" . $row['id'] . "</td>";
  echo "<td>" . $row['name'] . "</td>";
  echo "<td>" . $row['salary'] . "</td>";
  echo "<td>" . $row['job_role'] . "</td>";
  echo "</tr>";
}
echo "</table>";
mysqli_close($con);
?>
</body>
</html>
```

Example: Country State City Dropdown using Ajax in PHP MySQL

Step 1: Create Country State City Table

Step 2: Insert Data Into Country State City Table

Step 3: Create DB Connection PHP File

Step 4: Create Html Form For Display Country, State and City Dropdown

Step 5: Get States by Selected Country from MySQL Database in Dropdown List using PHP script

Step 6: Get Cities by Selected State from MySQL Database in DropDown List using PHP script

Advantages and Disadvantages of Ajax:

Advantages:

- Reduces the server traffic and increases the speed
- It is responsive and the time taken is also less
- Form validation
- Bandwidth usage can be reduced
- Asynchronous calls can be made; this reduces the time for data arrival.

Disadvantages:

- For security reasons, you can only access information from the web host that serves pages. Fetching information from other servers is not possible with Ajax.
- Most importantly, Ajax has a considerable dependency on JavaScript, so only browsers that support Javascripts or XMLHttpRequest can use pages with Ajax techniques
- Search Engines cannot index Ajax pages cannot be indexed by Google as well as other search engines
- The usage of Ajax can cause difficulties for your web pages to debug as well as make them prone to possible security issues in the future
- Users will find it challenging to bookmark a specific state of the application due to the dynamic web page eg. From the users' perspective, when you click the back button on the browser, you may not return to the previous state of the page but the entire page. This happens because the pages with successive Ajax requests are unable to register with the browser's history.