T. Y. B. Sc.
COMPUTER SCIENCE
SEMESTER-VI

NEW SYLLABUS
CBCS PATTERN

WEB TECHNOLOGIES-II

Dr. A. B. NIMBALKAR

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

Contents ...

■■■

CHAPTER 4

AJAX

Objectives...

- To Understand Ajax
- To Learn Ajax-PHP Framework
- To Study Handling XML Data Using PHP and Ajax

4.0 INTRODUCTION

- AJAX (Asynchronous JavaScript And XML) is a set of web development techniques that uses various web technologies on the client-side to create asynchronous web applications.
- With Ajax, web applications can send and retrieve data from a server asynchronously (in the background) without interfering with the display and behavior of the existing page.
- By decoupling the data interchange layer from the presentation layer, Ajax allows web pages and, by extension, web applications, to change content dynamically without the need to reload the entire page.
- AJAX just uses a combination of:
  - A browser built-in XMLHttpRequest object (to request data from a web server)
  - JavaScript and HTML DOM (to display or use the data)
- AJAX allows web pages to be updated asynchronously by exchanging 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.

4.1 OVERVIEW OF AJAX

[April 16, 18, 19, Oct. 17]

- Ajax is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and JavaScript.
- Ajax is not a technology but group of inter-related technologies as given below:
  1. **HTML/XHTML and CSS:** These technologies are used for displaying content and style. It is mainly used for presentation.
  2. **DOM:** It is used for dynamic display and interaction with data.

3. **XML or JSON:** For carrying data to and from server. JSON (Javascript Object Notation) is like XML but short and faster than XML.
4. **XMLHttpRequest:** For asynchronous communication between client and server. For more visit next page.
5. **JavaScript:** It is used to bring above technologies together. Independently, it is used mainly for client-side validation.

- Ajax is a technique for creating fast and dynamic web pages.
- 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 include Google Maps, Gmail, Youtube, and Facebook etc.

4.1.1 Advantages and Disadvantages of Ajax

Advantages of Ajax:

1. **Better interactivity:** Ajax allows easier and quicker interaction between user and website as whole pages are not reloaded for content to be displayed.
2. **Easier navigation:** Ajax applications on websites can be built to allow easier navigation to users in comparison to using the traditional back and forward button on a browser.
3. **Compact:** With Ajax, several multipurpose applications and features can be handled using a single web page. It just require a few lines of code.
4. **Backed by reputed brands:** Several complex web applications are handled using Ajax, Google Maps is the most impressive and obvious example.

Disadvantages of Ajax:

1. **Built on JavaScript:** A percentage of website surfers prefer to turn JavaScript functionality off on their browser making the Ajax application useless.
2. **Browser support:** All browsers do not support JavaScript or XMLHttpRequest object.
3. **Security and User privacy:** Not all concerns are addressed. Issues surrounding security and user privacy need to be considered when developing an Ajax application.
4. **Accessibility:** Because not all browsers have JavaScript or XMLHttpRequest object support, you must ensure that you provide a way to make the web application accessible to all users.
5. **Bookmark and Navigation:** Since, Ajax is used to asynchronously load bits of content into an existing page, some of the page information may not correspond to a newly loaded page. Browser history and bookmarks may not have the correct behavior since the URL was unchanged despite parts of the page being changed.

6. **Search engine:** Ajax applications are not searchable; however, it is possible to use Ajax features and elements within an application that is searchable.

4.2 AJAX WEB APPLICATION MODEL

- Before understanding Ajax, let's understand classic web application model and Ajax web application model first.

1. **Synchronous (Classic Web Application Model):**

- The nature of interaction between the client and the server is of start-stop-start-stop (i.e. click-wait)
- The browser response to the user action by discarding the current HTML page. The request is sent to the web server.
- When the server completes the processing of request, it returns the response page to the Web browser.
- Browser refreshes the screen and displays the new HTML page. This is called as synchronous request response model.

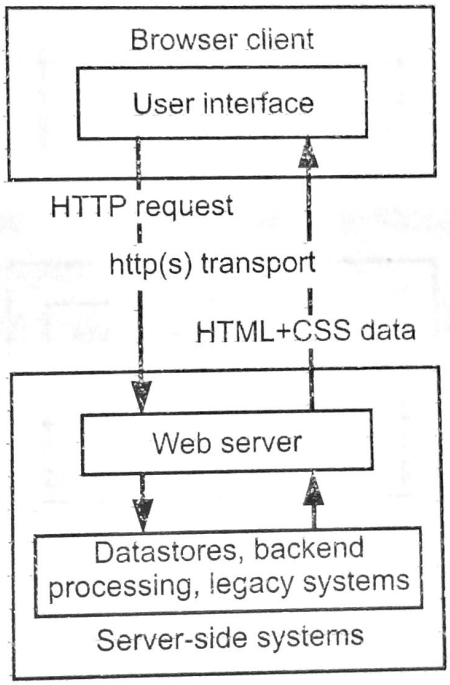


Fig. 4.1: Synchronous Web Application Model

- This approach makes a lot of technical sense, but it doesn't make for a great user experience. While the server is doing its thing, but user is waiting.
- And at every step in a task, the user waits some more. In fact, why should the user see the application go to the server at all?

2. **Asynchronous (Ajax Web-Application Model):**

- The intermediary layer (i.e. Ajax Engine) is introduced between the user and the Web server.
- The Web page sends its requests using JavaScript. The request is done asynchronously, meaning that code execution does not wait for response.

- The server response comprises of data and not the presentation. The Ajax engine can understand and interpret the data.
- Most of the page does not change, only parts of the page that need to change are updated.
- The JavaScript dynamically updates the web page, without redrawing everything. For the Web server nothing has change; it still responds to each request.
- This way you never have to wait around. The Ajax engine, irrespective of the server, does asynchronous communication.

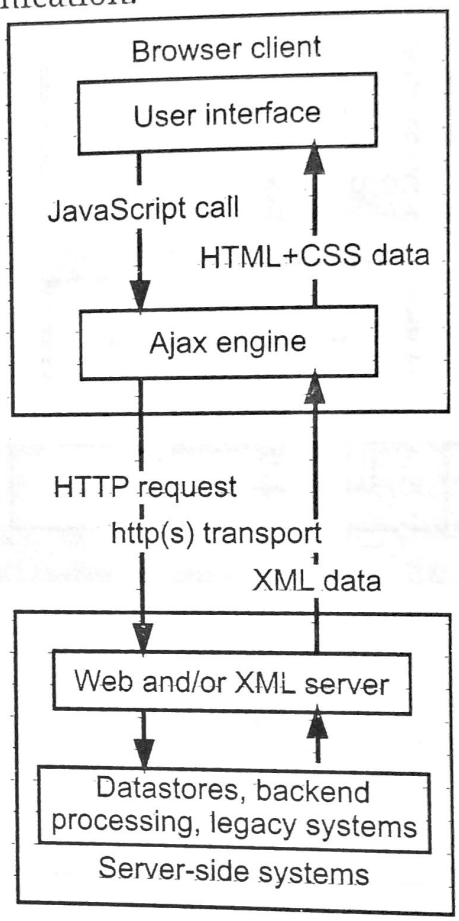


Fig. 4.2: Asynchronous Web Application Model

Understanding XMLHttpRequest: **[April 16, 17, 19, Oct. 18]**

- The XMLHttpRequest object is the key to Ajax. The XMLHttpRequest is used for asynchronous communication between client and server.
- The XMLHttpRequest object can send HTTP request, and receive responses.
- The XMLHttpRequest object transfer the XML and other text data to and from the Web server by using HTTP.
- The XMLHttp Object can be used in calling the web page in either synchronous or asynchronous mode.
- It establishes an independent connection channel between a webpage's Client-Side and Server-Side.

- It performs following operations:
  1. Sends data from the client in the background,
  2. Receives the data from the server, and
  3. Updates the webpage without reloading it.

XMLHttpRequest Object's Properties:

1. **onreadystatechange**: An event handler for an event that fires at every state change. This property sets the method to be called on every state change.
2. **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.

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

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

   (iii) **readyState = 2** After you have called send().

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

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

3. **responseText**: Returns the response as a string.
4. **responseXM:** 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.
5. **status:** Returns the status as a number (e.g., 404 for "Not Found" and 200 for "OK").
6. **statusText:** Returns the status as a string (e.g., "Not Found" or "OK").

XMLHttpRequest Methods:

1. **abort():** This method is used to cancel the current XMLHttpRequest and reset the object to be uninitialized state.
2. **getAllResponseHeaders():** Returns the complete set of HTTP headers as a string.

3. **getResponseHeader(headerName):** Returns the value of the specified HTTP header.
4. **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.
5. **send(content):** Sends the HTTP request to the server and receives the response when the readyState value is 1.
6. **setRequestHeader(label, value):** Adds a label/value pair to the HTTP header to be sent.

Creating the XMLHttpRequest Object:

- The XMLHttpRequest is implemented in different ways by the browsers. In Internet Explorer 6 and older, XMLHttpRequest is implemented as an ActiveX control and you instantiate it like this:

```
xmlhttp = new ActiveXObject("Microsoft.XMLHttp");
```

- For the other web browsers, XMLHttpRequest is a native object, so you create instances of it like this:

```
xmlhttp = new XMLHttpRequest();
```

- The following code is used to create XMLHttpRequest object:

```
function loadXMLDoc()
{
  // will store the reference to the XMLHttpRequest object
  var xmlhttp;
  if (window.XMLHttpRequest)
  {
    // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest(); // try to create XMLHttpRequest object
  }
```

```
    else
    {
        // code for IE6, IE5
        // try to create XMLHttpRequest object
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
}
```

How XMLHttpRequest Object Works in Synchronous Pages Pattern:

- The XMLHttpRequest object works in the following pattern:
  - Create the object.
    For example: `xmlHttp = new XMLHttpRequest();`
  - Create the request.
    For example: `xmlHttp.open ("GET",url,false);`
  - Send the request.
    For example: `xmlHttp.send (null);`
  - Hold the processing until you get the response i.e. get the response by responseText property,
    For example: `var xmlHttp = xmlHttp.responseText;`

How XMLHttpRequest Object Works in Asynchronous Pages Pattern:

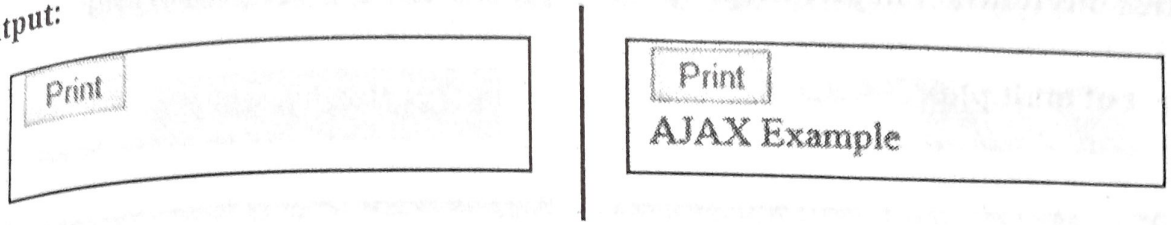
- The XMLHttpRequest object works in the following pattern:
  - Create the object.
  - Set the readystatechange event to trigger the specific function.
  - Check the readyState property, to see if data is ready. If it is not, then check it again after an interval.
  - Open request.
  - Send request.
  - Continue the processing. Interruption is done only when the response is received.
- The above steps are performed by the following code:

```
xmlHttp = new XMLHttpRequest();
xmlHttp.onreadystatechange = stateChanged
xmlHttp.open ("GET", url, true);
xmlHttp.send (null);
var xmlHttp = xmlHttp.responseText;
function stateChanged()
{
  if (xmlHttp.readyState == 4)
  {
     var objXML = xmlHttp.responseXML;
  }
}
```

- **Example:** Ajax program to read a text file and print the contents of the file when the user clicks on the Print button.
- Create a text file named "a.txt" and write text "Ajax Example" into it.
- Create an HTML file with the following code:

```
<html>
   <head>
      <script>
         function loadXMLDoc()
         {
            var xmlhttp;
            if (window.XMLHttpRequest)
            {// code for IE7+, Firefox, Chrome, Opera, Safari
               xmlhttp=new XMLHttpRequest();
            }
            else
            {// code for IE6, IE5
               xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
            }
               xmlhttp.open("GET","a.txt",true);
               xmlhttp.send(null);
               xmlhttp.onreadystatechange=function()
            {
               if (xmlhttp.readyState==4 && xmlhttp.status==200)
               {
               document.getElementById("myDiv").innerHTML=
                                                   xmlhttp.responseText;
                  }
               }
            }
         </script>
   </head>
   <body>
      <button onclick="loadXMLDoc()">Print</button>
      <div id="myDiv"></div>
   </body>
</html>
```

Output:



- The above program will start from HTML body tag which contain a button tag and a div tag, when user click on the 'Print' button, the JavaScript function loadXMLDoc() will be called.
- Inside the function the Ajax works and it send a GET request for the file 'a.txt', and if request is completed then the value of the readyState property is equals to 4 and status is 200, so it fetches text from the file using property responseText and display the text in place of div tag.

4.3 AJAX - PHP FRAMEWORK [Oct. 14]

- The PHP-frameworks gives different ways to affect the client view in both template based view (uses technologies like JSP, ASP and RHTML) and server-side technologies (like ASP.NET and JSP).
- This functionality becomes popular in the server-side Ajax packages, because it automatically create JavaScript code for us.
- There are various PHP frameworks available, like Ajax Core, CakePHP, Xajax, Sajax, XOAD, Zephyr, Feather Ajax 1.1 and Tigermouse to integrate Ajax with PHP.
- These frameworks support Model, View and Controller (MVC) architecture. They reduces the writing of same code and common function repeatedly.
- The Sajax is an Ajax based framework which generates Ajax -enabled JavaScript from many server side languges like PHP, ASP, ColdFusion, Io, Lua, Perl, Python and Ruby. This Sajax bridge execute server side code and uses Object Remoting technique.
- The object brokers enable remote exchange and the client-side methods are tied to the server side object. There are network round trips involved and messages are sent via service oriented frameworks.

Sajax Framework Example:

- In this example, JavaScript function generates from PHP function on server side that manipulates data and returns result to other javascript function on client side.
- The Home page of mult.php has three text fields where user can enter two numbers in first two text field and in third text field, result is displayed. There is one submit button in home page. We have to include Sajax.php in mult.php.
- We need to setup Sajax using Sajax_init() method. This file contains function mult for multiplication of two numbers.

- To access this function in JavaScript by another name, we need to export this method, like x_mult.

The part of mult.php:

```
<?
    require ("Sajax.php");
        function mult ($no1, $no2)
        {
            return $no1 * $no2;
        }
        Sajax_init ();
            Sajax_export ("mult");
            Sajax_handle_client_request ();
        //it connect mult function with Sajax and generate JavaScript?>
```

- The generated JavaScript code can be embed in web page using PHP function `Sajax_show_javascript ( );`
- After Clicking on Submit button, it invokes x_mult function, which in turn calls mult function on server side.

The part of mult.php:

```
<script>
<?
    Sajax_show_javascript ();
?>
    function show_results (result)
    {
        docment.getElementById("result").value=result;
    }
        function do_add ()
        {
            var no1, no2;
            no1 = document.getElementById("No1").value;
            no2 = document.getElementById("No2").value;
            x_mult (no1, no2, show_results);
        }
</script>
```

- The x_mult uses three argument, first is value, second is value and third is a function name which will display the result of multiplication.

4.4 PERFORMING AJAX VALIDATION

- The validation may done for checking integer, email id or phone number etc. The first page for application displays the text field to enter the username.
- It has JavaScript code to create the XMLHttpRequest object. The validate method sends request to validate the PHP page with parameter name.
- When status of response is OK, the element having id res, is populated with the text response received from server. When new request comes in the abort, method exists from the previous request.
- **Example:** Ajax program to carry out validation for a username entered in textbox. If the textbox is blank, print 'Enter username'. If the number of characters is less than three, print 'Username is too short'. If value entered is appropriate the print 'Valid username'.

a.html

```
<html>
   <head>
      <script type="text/javascript">
         var xmlHttp = false;
         function validate(name)
         {
            // alert(str);
            if(window.XMLHttpRequest)
         {
            xmlHttp = new XMLHttpRequest();
         }
         else if(window.ActiveXObject)
         {
            xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
         }
            // alert("c3");
            if(xmlHttp==null)
         {
            alert("Browser does not support HTTP request");
            return;
         }
            //xmlHttp.abort(); //alert(name);
            xmlHttp.open("GET", "validation.php?name=" + name, true);
            xmlHttp.onreadystatechange=stateChanged
            xmlHttp.send(null);
         }
```

```
            function stateChanged()
            {//alert(xmlHttp.readyState);
               if(xmlHttp.readyState==4)
               {
                  document.getElementById("res").innerHTML=xmlHttp
                                                          .responseText;
               }
            }
         </script>
      </head>
      <body>
         <form>
            Username: <input type="text" name="name"
                                          onKeyUp="validate(this.value)" />

            <div id="res"></div>
         </form>
      </body>
   </html>
```

- The Ajax.html and validate.php files are stored into the server www directory and executed in browser.

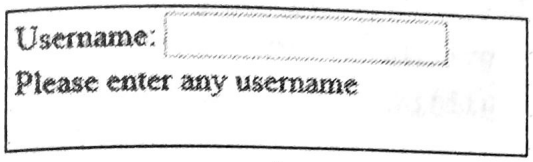
  `http://localhost/a.html`

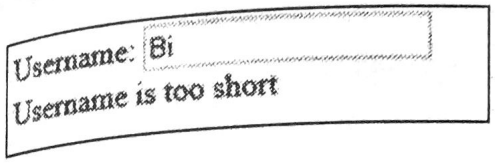
- The a.html file calls validate.php where validation is done. If name is blank or name is less than 3 characters or if name is already exists, the corresponding messages are given.

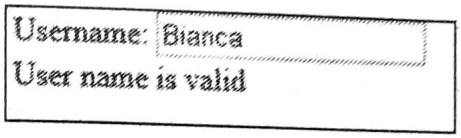
validate.php

```
<?php
   function validate($name)
   {
      if($name == '')
         return 'Please enter any username';
      if(strlen($name) < 3)
         return 'Username is too short';
      if(strlen($name) > 10)
         return 'Username is too long';
      return 'User name is valid';
   }
   echo validate($_GET['name']);
?>
```

Output:







4.5 HANDLING XML DATA USING PHP AND AJAX

- XML files are used to store data using our own defined tags. The following example shows how to get the data from XML file using Ajax.
- The index.html file use choosebook.js JavaScript file. This form shows the Book title to the user and when user select the title from list box, the sendTitle method from choosebook.js is called.

index.html

```
<html>
   <head>
      <script type="text/javascript">
         var xmlHttp;
         function sendTitle(str)
         {
            if(window.XMLHttpRequest)
            {
               xmlHttp = new XMLHttpRequest();
            }
            else if(window.ActiveXObject)
            {
               xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
            }
               if(xmlHttp==null)
               {
                  alert("Browser does not support HTTP request");
                  return;
               }
               var url="getbook.php";
               url=url+"?q="+str;
```

```
                xmlHttp.onreadystatechange=stateChanged
                xmlHttp.open("GET",url,true);
                xmlHttp.send(null);
                }
            function stateChanged()
            {
                if(xmlHttp.readyState==4)
                {
                    document.getElementById("res").
                                innerHTML=xmlHttp.responseText;
                }
            }
        </script>
    </head>
    <body>
        <form>
            List of book titles:
            <select name="titles" onChange="sendTitle(this.value)">
                <option value="AJAX ">AJAX </option>
                <option value="Java2">Java2</option>
                <option value="HTML5">HTML5</option>
                <option value="PHP6">PHP6</option>
            </select>
            </form>
        <div id="res">Book details will be given here</div>
    </body>
</html>
```

- The booksdata.xml file stores the Book's details like Title, Author, Year and price sub elements.

bookstore.xml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<bookstore>
    <book>
        <title>AJAX </title>
        <author>author1</author>
        <year>2000</year>
```

```
    <price>250</price>
  </book>
  <book>
    <title>Java2</title>
    <author>author2</author>
    <year>2005</year>
    <price>600</price>
  </book>
  <book>
    <title>HTML5</title>
    <author>author3</author>
    <year>2010</year>
    <price>300</price>
  </book>
  <book>
    <title>PHP6</title>
    <author>author4</author>
    <year>2013</year>
    <price>400</price>
  </book>
</bookstore>
```

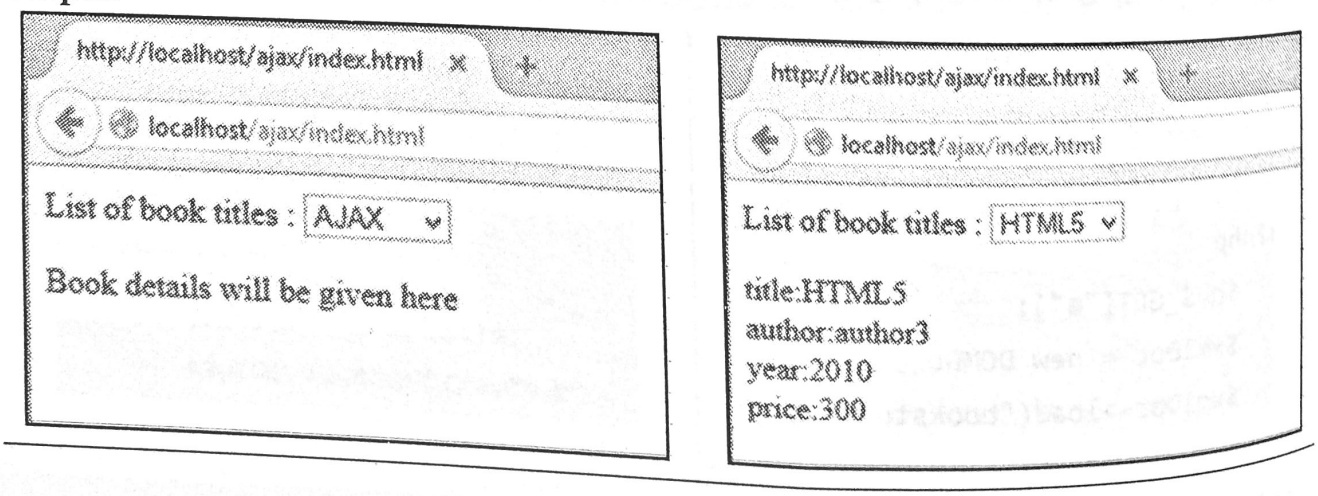
- The XMLHttpRequest method creates URL to request getbook.php file. This URL has parameter q which is used to initialize the value of list box and then sends the request to getbook.php page. After completion of response by the server, the stateChanged method is called.
- The server page getbook.php create XML DOMDocument object user select an Option. Then booksdata.xml file is loaded.
- The Title send from HTML form search in booksdata.xml file. This way the details of the selected book are displayed.

getbook.php

```
<?php
  $q=$_GET["q"];
  $xmlDoc = new DOMDocument();
  $xmlDoc->load("bookstore.xml");
  $x=$xmlDoc->getElementsByTagName('title');
```

```
    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;
          }
       }
    }
    $book=$y->childNodes;
    for($i=0;$i<$book->length;$i++)
    {
       //Process only element nodes
       if ($book->item($i)->nodeType==1)
       {
          echo($book->item($i)->nodeName);
          echo(":");
          echo($book->item($i)->childNodes->item(0)->nodeValue);
          echo("<br/>");
       }
    }
?>
```

Output:



4.6 CONNECTING DATABASE USING PHP AND AJAX

- Storing the data into database and retrieving it from database, we can do this using PHP and PostgreSQL.
- Consider an example of employee database table. From index.html form, we can select the employee name using combo box and display its record details.
- Create 'Employees' table in PostgreSQL, with fields Name, Age, City, Designation. Insert the names as per db.html with all details.

db.html

```
<html>
   <head>
      <script type="text/javascript">
         var xmlHttp;
         function sendEmpID(str)
         {
            // alert(str);
            if(window.XMLHttpRequest)
            {
               xmlHttp = new XMLHttpRequest();
            }
            else if(window.ActiveXObject)
            {
               xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
            }
               // alert("c3");
               if(xmlHttp==null)
               {
                  alert("Browser does not support HTTP request");
                  return;
               }
               var url="getemployee.php";
               url=url+"?q="+str + "&sid=" + Math.random();
               xmlHttp.onreadystatechange=stateChanged
               // alert(url);
               xmlHttp.open("GET",url,true);
               xmlHttp.send(null);
            }
```

```
            function stateChanged()
            {//alert(xmlHttp.readyState);
                if(xmlHttp.readyState==4)
                {
                    document.getElementById("emp").
                                        innerHTML=xmlHttp.responseText;
                }
            }
        </script>
    </head>
    <body>
        <form>
            Employee List:
            <select name="names" onChange="sendEmpID(this.value)">
                <option value="1">Mahesh</option>
                <option value="2">Sachin</option>
                <option value="3">Tejas</option>
                <option value="4">Bhavesh</option>
            </select>
        </form>
        <div id="emp">Employee info will be listed here</div>
    </body>
</html>
```

- The method sendEmpID() sends asynchronous request to getemployee.php file. The request URL has parameter q to store id. It sends the request to getemployee.php with parameter.
- After selecting the employee name, it sends the id as query parameter to getemployee.php page, this page establish the connection to PostgreSQL server. After successful connection, it will retrieve data and using HTML table, the fetched values are inserted into corresponding cell or row.

getemployee.php

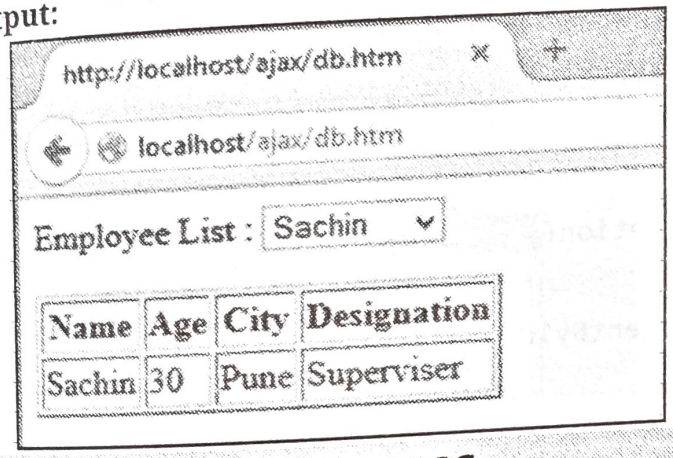
```
<?php
    q=$_GET["q"];
        $conn = pg_pconnect("dbname=test") or die("An error occurred.");
         $result = pg_query($conn, "SELECT * FROM employee WHERE id = ". $q)
                                            or die("An error occurred.");

        echo "<table border='1'>
        <tr>
            <th>Name</th>
            <th>Age</th>
```

```
        <th>City</th>
        <th>Designation</th>
    </tr>";
while($row = pg_fetch_array($result, NULL, PGSQL_ASSOC))
{
    echo "<tr>";
    echo "<td>". $row['name']. "</td>";
    echo "<td>". $row['age']. "</td>";
    echo "<td>". $row['city']. "</td>";
    echo "<td>". $row['designation']. "</td>";
    echo "</tr>";
}
    echo "</table>";
    pd_close($conn);
?>
```

- We execute this using http://localhost index.html.

Output:



ADDITIONAL PROGRAMS

Program 1: Program to display list of book stored in an array on clicking ok button.

Book_list.php

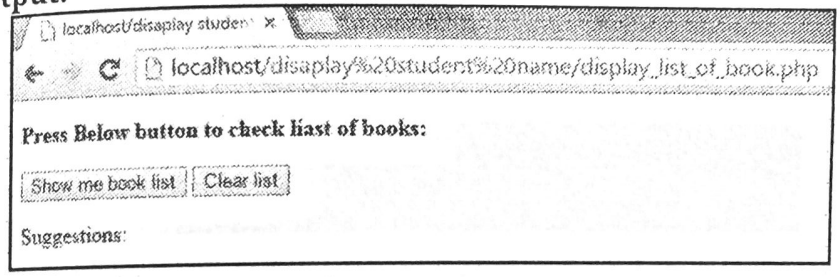
```
<html>
    <head>
        <script>
            function showMatch()
            {
            //var str = document.getElementById("search_string");
            /*if (str.length==0)
            {
                document.getElementById("txtHint").innerHTML="";
                return;
            }*/
```

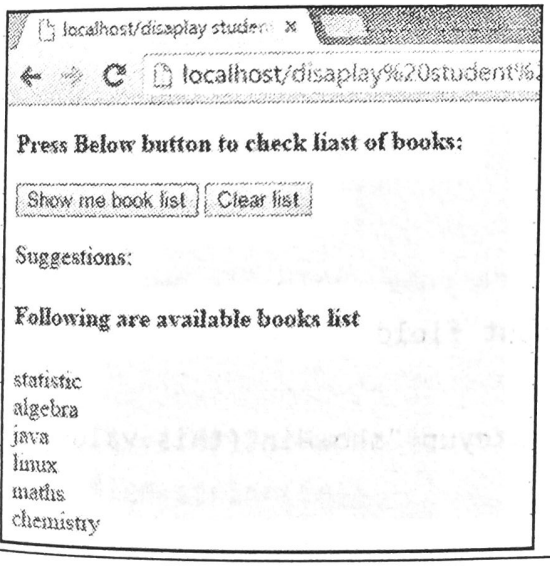
```
        if (window.XMLHttpRequest)
        {  // code for IE7+, Firefox, Chrome, Opera, Safari
           xmlhttp=new XMLHttpRequest();
        }
        else
        {  // code for IE6, IE5
           xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
        }
           xmlhttp.onreadystatechange=function()
        {
           if(xmlhttp.readyState==4 && xmlhttp.status==200)
           {
              document.getElementById("txtHint").innerHTML
                                          =xmlhttp.responseText;
           }
         }
           xmlhttp.open("GET","getmatch.php",true);
           xmlhttp.send();
         }
           function clear_suggetion()
           {
              document.getElementById("txtHint").innerHTML="";
           }
      </script>
   </head>
   <body>
      <p><b>Press Below button to check liast of books:</b></p>
      <form>
         <input name="submit" type="button" value="Show me
                                    book list" onclick="showMatch()" />
         <input name="submit" type="button" value="Clear list"
                                          onclick="clear_suggetion()" />
      </form>
      <p>Suggestions: <span id="txtHint"></span></p>
   </body>
</html>
```

Getmatch.php

```
<?php
    //print_r($_GET);
    // this is array to search
    $arry_tosearch = array("statistic", "algebra",
                                    "java", "linux", "maths", "chemistry");
    // print above array
    //echo count($arry_tosearch);
    echo "<br><h4>Following are available books list</h4>";
    for($i=0 ; $i<count($arry_tosearch); $i++)
    {
        echo  "".$arry_tosearch[$i],'<br>';
    }
?>
```

Output:





Program 2: Program to search student name according the character typed and display list using array.

New.php

```
<html>
    <head>
        <script>
```

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

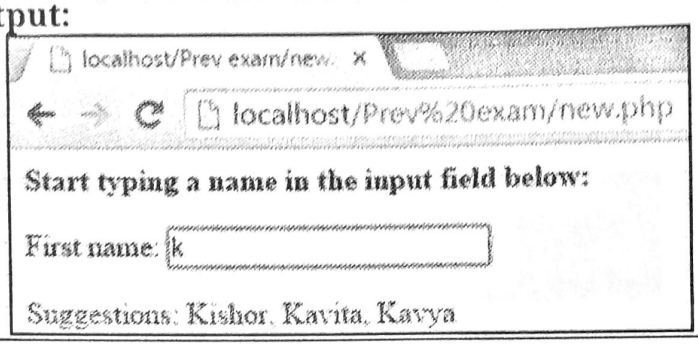
Gethint.php

```
<?php
    // Array with names
    $a[] = "Amit";
```

```
$a[] = "Bob";
$a[] = "Chinmay";
$a[] = "Deepa";
$a[] = "Esha";
$a[] = "Faiz";
$a[] = "Gautam";
$a[] = "Harshal";
$a[] = "Ishant";
$a[] = "Jyotsana";
$a[] = "Kishor";
$a[] = "Lokesh";
$a[] = "Nina";
$a[] = "Omkar";
$a[] = "Pritam";
$a[] = "Amey";
$a[] = "Rahul";
$a[] = "Chaitali";
$a[] = "Dushyant";
$a[] = "Kavita";
$a[] = "Kavya";
$a[] = "Sunil";
// 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;
?>
```

Output:



Program 3: Program to program to display list of games stored in an array on clicking ok button.

Disp_list.php

```
<html>
    <head>
        <script>
            function showMatch()
            {
            //var str = document.getElementById("search_string");
            /*if (str.length==0)
            {
                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(xmlhttp.readyState==4 && xmlhttp.status==200)
            {
                document.getElementById("txtHint").innerHTML=
                                                    xmlhttp.responseText;
            }
            }
                xmlhttp.open("GET","getmatch.php",true);
                xmlhttp.send();
            }
                function clear_suggetion()
                {
                    document.getElementById("txtHint").innerHTML="";
                }
    </script>
</head>
<body>
    <p><b>Press Below button to check list of games:</b></p>
    <form>
        <input name="submit" type="button" value="Show me game list"
                                                onclick="showMatch()" />
        <input name="submit" type="button" value="Clear list"
                                            onclick="clear_suggetion()" />
    </form ons>
    <p>Suggestions: <span id="txtHint"></span></p>
</body>
</html>
```

Getmatch.php

```
<?php
    //print_r($_GET);
    // this is array to search
    $arry_tosearch = array("hocky", "soccer", "cricket", "table tennis",
                                                "badminton", "chess");
    // print above array
    //echo count($arry_tosearch);
```

```
    echo "<br><h4>Following are available game list</h4>";
    for($i=0 ; $i<count($arry_tosearch); $i++)
    {
        echo  "".$arry_tosearch[$i],'<br>';
    }
?>
```

Output: