Unit 6

Ajax and XML

1. Introduction

AJAX stands for "Asynchronous JavaScript and XML". It is a group of existing technologies (i.e. HTML, CSS, JavaScript, XML, etc.) which come together to build modern web applications. With AJAX, a client (i.e. browser) communicates with a web server and asks for data. Then, it processes the server's response and makes changes to the page without fully reloading it.

i. "Asynchronous" means that when a client requests data from a web server, it doesn't freeze until the server replies. On the contrary, the user can still navigate the pages. As soon as the server returns a response, a relevant function manipulates the returned data behind the scenes.

ii. "JavaScript" is the language which instantiates an AJAX request, parses the corresponding AJAX response, and finally updates the DOM.

iii. A client uses the XMLHttpRequest or XHR API to make a request to a server. API(Application Programming Interface) is a set of methods which specify the rules of communication between the two interested parties.

2. Understanding the Basics of JavaScript

1. JavaScript is a Scripting Language.
2. A scripting language is a lightweight programming language.
3. JavaScript is programming code that can be inserted into HTML pages.
4. JavaScript inserted into HTML pages, can be executed by all modern web browsers.
5. // is the JavaScript comment symbol. Multiple line comments are given by /*and */
6. Curly brackets { } are used to indicate a block of code.
7. A semicolon is used to define end of statement which is optional.
8. The HTML <script> tag is used to insert a JavaScript into an HTML page. All JavaScript coding is written between the opening and closing script tags.
9. *Example,*

```
<html>
<body>
<script type="text/javascript">
     document.write("This is my first JavaScript message");
</script>
</body>
</html>
```

The command document.write is the standard JavaScript command for writing the output to the page.

2.1 General Syntactic Characteristics

JavaScript can be placed in the <body> and the <head> sections of an HTML page.

The <script> Tag

In HTML, JavaScript code must be inserted between <script> and </script> tags.

Example

```
<script>
    document.getElementById("demo").innerHTML = "My First JavaScript";
</script>
```

JavaScript Functions and Events

A JavaScript function is a block of code that will be executed when "someone" calls it.

A function is written as a code block (inside curly { } braces),

preceded by the function keyword:

```
function functionname()
{
  some code to be executed
}
```

Example

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
{
  alert("Hello World!");
}
</script>
</head>
<body>
  <button onclick="myFunction()">Try it</button>
</body>
</html>
```

JavaScript in <head> or <body>

We can place any number of scripts in an HTML document.

Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

JavaScript in <head>

In this *example*, a JavaScript function is placed in the <head> section of an HTML page.

The function is invoked (called) when a button is clicked:

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
{
    document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>
<body>
  <h1>My Web Page</h1>
  <p id="demo">A Paragraph</p>
  <button type="button" onclick="myFunction()">Try it</button>
</body>
</html>
```

JavaScript in <body>

In this *example*, a JavaScript function is placed in the <body> section of an HTML page.

The function is invoked (called) when a button is clicked:

```
<!DOCTYPE html>
<html>
<body>
  <h1>My Web Page</h1>
  <p id="demo">A Paragraph</p>
  <button type="button" onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
   document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</body>
</html>
```

JavaScript Display Possibilities

JavaScript can "display" data in different ways:

1. Writing into an alert box, using window.alert().
2. Writing into the HTML output using document.write().
3. Writing into an HTML element, using innerHTML.
4. Writing into the browser console, using console.log().

2.2 Data Types Supported by JavaScript

A variable in JavaScript can contain any data. A variable can at one moment be a string and later receive a numeric value:

```
var x;              // Now x is undefined
var x = 5;          // Now x is a Number
var x = "John";     // Now x is a String
```

Programming languages that allow such things are called "dynamically typed", meaning that there are data types, but variables are not bound to any of them.

Following are the basic data types in JavaScript.

i. Number
ii. String
iii. Boolean
iv. Undefined
v. Null
vi. Reference data types-object and function

Type	Example
Numbers	Any number, such as 14,29 or 54e7
Strings	"Hello!" or "World" or "empty string like "
Boolean	Either true or false
Null	A special keyword for exactly that – the null value (that is, nothing)

1. **JavaScript Numbers:** JavaScript has only one type of numbers. Numbers can be written with, or without decimals:

```
var x1=34.00; //Written with decimals
var x2=34; //Written without decimals
```

2. **JavaScript Strings:** A string can be any text inside quotes. we can use single or double quotes:

```
var carname="Volvo XC60";
var carname='Volvo XC60';
```

3. **JavaScript Booleans:** Booleans can only have two values: true or false.

```
var x=true;
var y=false;
```

4. **JavaScript Arrays:** The following code creates an Array called cars:

```
var cars=new Array();
cars[0]="Saab";
```

```
cars[1]="Volvo";
cars[2]="BMW";
```

or (condensed array)

```
var cars=new Array("Saab","Volvo","BMW");
```

or (literal array)

```
var cars=["Saab","Volvo","BMW"];
```

5. **JavaScript Objects**

```
var person={firstname:"John", lastname:"Doe", id:5566};
```

The object (person) in the *example* above has 3 properties: firstname, lastname, and id.

Spaces and line breaks are not important. the declaration can span multiple lines:

```
var person
{
  firstname: "John",
  lastname: "Doe",
  id: 5566
};
```

we can address the object properties in two ways:

Example

```
name=person.lastname;
name=person["lastname"];
```

6. **Undefined and Null:** Undefined is the value of a variable with no value.

Variables can be emptied by setting the value to null;

Example

```
cars=null;
person=null;
```

3. AJAX

AJAX is not a programming language but a technique for creating better, faster and more interactive web applications. AJAX uses asynchronous data transfer between browsers and web servers, allowing web pages to request small bits of information from the Web server instead of whole pages.

Definition

Ajax stands for Asyncronous Javascript and XML. Basically AJAX makes use of javascript-based XMLHttpRequest object to fire request to the web server asynchronously-or without having to refresh the page. *Figure* below illustrates the difference between traditional and Ajax-based request/response models. By making use of XMLHttpRequest, web applications can garner/send information to the server, have the server to do any processing that needs to be handled, and then change aspects of the web page dynamically without the user having to move pages.

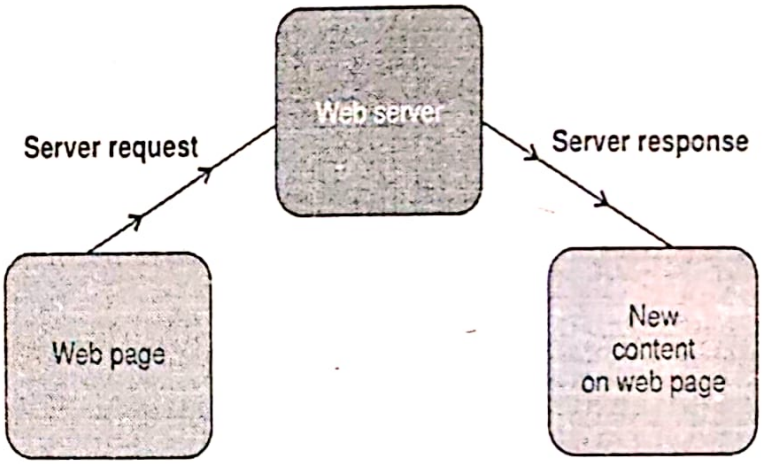


Figure 6.1

1

Apr. 2019 – *1M*
List the example of AJAX.

For example: If you are using an income tax return calculator form to deduce the amount of money from your account-not a big task for the scripting language. Normally such an application will be to simply fill the form and press submit button and wait for response to come back. From there you could redo the entire thing, testing with new financial *figures*.

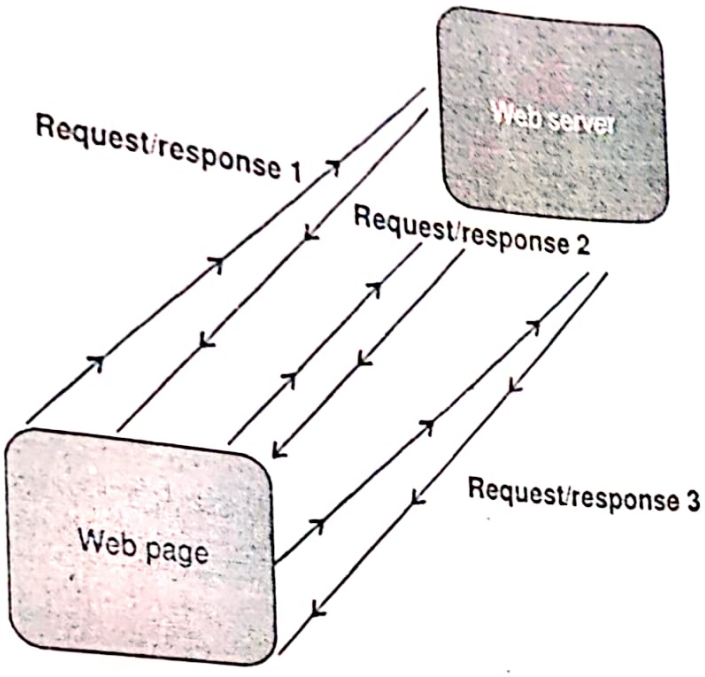


Figure 6.2

Application of Ajax

With a Javascript-based Ajax solution, however we would click the submit button, and while we remain on the same page the server will do the calculations and return the value of amount to be paid in front of our eyes. We could change the values in the formula and immediately see the differences.

The *examples* where AJAX can make a difference are endless. Following are live and popular *examples*:

i. **Google Suggests** help us with Google searches. The functionality is pretty spectacular, similar functionality is offered by **Yahoo! Instant search.**

ii. **Gmail** is very popular and doesn't need any introduction. Other web based email services **Yahoo mail** and **Hotmail** have followed the trend and offered Ajax based functionality.

iii. **Google Maps, YahooMaps** are few more applications.

Following are the advantages and disadvantages of ajax

Advantages

i. Reduced page bandwidth.

ii. Load page data on demand.

iii. Service-based approach to web development.

iv. Rich user experience.

Disadvantages

i. AJAX is dependent on Javascript. If there is some Javascript problem with the browser or in the OS, Ajax will not support.

ii. Ajax can be problematic in Search engines as it uses Javascript for most of its parts.

iii. Source code written in AJAX is easily human readable. There will be some security issues in Ajax.

iv. Debugging is difficult.

v. Increases size of the requests.

vi. Slow and unreliable network connection.

vii. Problem with browser back button when using AJAX enabled pages.

The differences between AJAX and JavaScript are as follows:

	AJAX	Javascript
i.	AJAX sends request to the server and does not wait for the response. It performs other operations on the page during that time.	JavaScript makes a request to the server and waits for response.
ii.	AJAX does not require the page to refresh for downloading the whole page.	JavaScript manages and controls a Web page after being downloaded.
iii.	AJAX minimizes the overload on the server since the script needs to request once.	JavaScript posts a request that updates the script every time.

4. AJAX Web Application Model

An Ajax application eliminates the start-stop-start-stop nature of interaction on the Web by introducing an intermediary an Ajax engine between the user and the server.

Working of AJAX

The AJAX-enabled applications, on the other hand, rely on a new asynchronous method of communication between the client and the server.

It is implemented as a JavaScript engine that is loaded on the client during the initial page load. From there on, this engine serves as a mediator that sends only relevant data to the server as XML and subsequently processes server response to update the relevant page elements. Below is a diagram of the complete lifecycle of an AJAX.

In contrast, the traditional synchronous (postback-based) communication would require a full page reload every time data has to be transferred to/from the server.

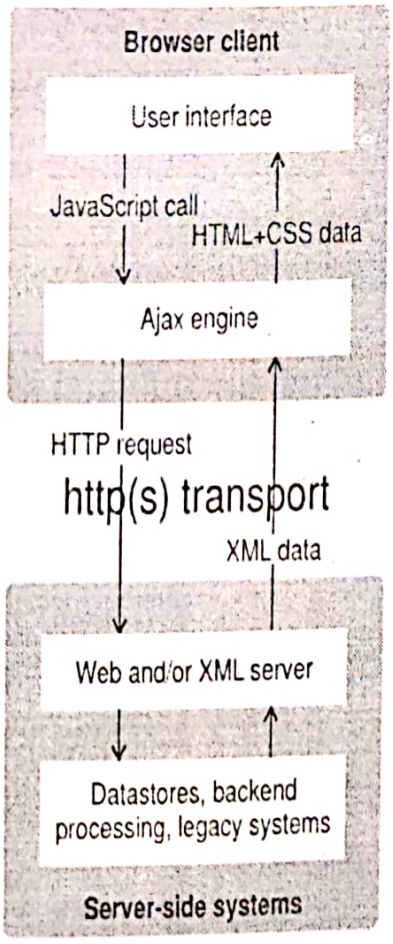


Figure 6.3

Figure shows how the intermediately layer is introduced between the user and the web server.

1. The web page sends the request using a java script function.
2. The JS code makes a request to the server.
3. The server response comprises of data and not the presentation which implies that the data required by the page is provided by the server as the response, and the presentation is implemented on the data with the help of mark-up language.

Every user action that normally would generate an HTTP request takes the form of a JavaScript call to the Ajax engine instead.

Creating a Sample AJAX Application

```
<html>
<head>
<script type="text/javascript">
function loadXMLDoc()
{
  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("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
  xmlhttp.open("GET","ajaxinfo.txt",true);
  xmlhttp.send();
}
```

```
</script>
</head>
<body>
<div id="myDiv"><h2>First Ajax Example to Change Text</h2></div>
<button type="button" onclick="loadXMLDoc()">Change text</button>
</body>
</html>
```

Which will display the following window

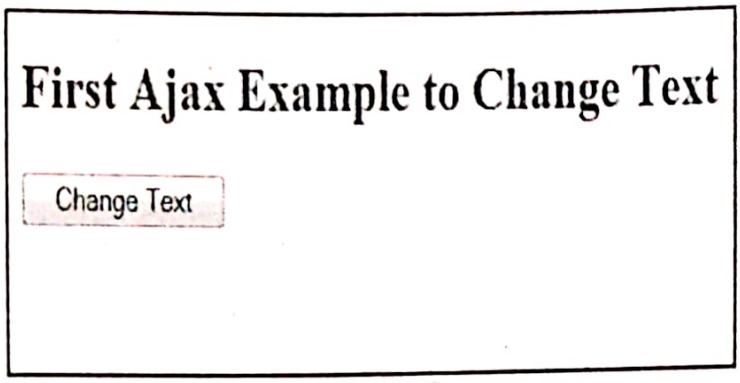


Figure 6.4

On click of the button "Change Text" it will call the "ajaxinfo.txt" page, which will display the following information.

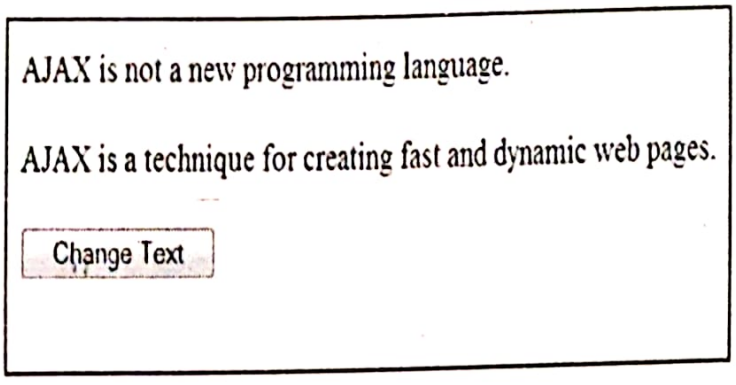


Figure 6.5

Consider another *example* which loads an xml file data in an ajax program.

```
<html>
<head>
<script type="text/javascript">
function loadXMLDoc(url)
{
  if(window.XMLHttpRequest)
  {
    // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
  }
  xmlhttp.onreadystatechange=function()
  {
    if(xmlhttp.readyState==4 && xmlhttp.status==200)
    {
      document.getElementById('A1').innerHTML=xmlhttp.status;
      document.getElementById('A2').innerHTML=xmlhttp.statusText;
      document.getElementById('A3').innerHTML=xmlhttp.responseText;
    }
  }
  xmlhttp.open("GET",url,true);
  xmlhttp.send();
}
</script>
</head>
<body>
<h2>Retrieve data from XML file</h2>
<p><b>Rollno:</b><span id="A1"></span></p>
<p><b>Name:</b><span id="A2"></span></p>
<p><b>Address:</b><span id="A3"></span></p>
  <button onclick="loadXMLDoc('stud.xml')">Get the XML data</button>
</body>
</html>
```

5. AJAX-PHP Framework

1

Apr. 2019 – 5M
Write a note on an AJAX PHP framework.

All PHP frameworks provide different ways to affect the client view in both template view rendering technologies (like JSP, ASP and RHTML) and server side technologies with self rendering technologies. These frameworks now support the creation of javascript for clients to have AJAX affects.

Sajax is an open source tool to make programming websites using the Ajax framework -also known as XMLHTTPRequest or remote scripting - as easy as possible. Sajax makes it easy to call PHP, Perl or Python functions from the webpages via JavaScript without performing a browser refresh.

For sajax functionality we have to include Sajax.php in mul.php and need to set up sajax using `sajax_init()` method.

Handle_client_request method connects `multiply()` with sajax and generates javascript.

Mul.php

```
<?
   require("Sajax.php");
   function multiply($x, $y)
   {
      return $x * $y;
   }
   sajax_init();
   sajax_export("multiply");
   sajax_handle_client_request();
?>
<html>
<head>
<title>Multiplier</title>
<script>
<?
   sajax_show_javascript();
?>
function multiply_cb(z)
{
   document.getElementById("z").value = z;
}
   function multiply()
```

```
{
  // get the folder name
  var x, y;
    x = document.getElementById("x").value;
    y = document.getElementById("y").value;
    x_multiply(x, y,multiply_cb);
}
</script>
</head>
<body>
  <input type="text" name="x" id="x" value="2" size="3">
      *
  <input type="text" name="y" id="y" value="3" size="3">
      =
  <input type="text" name="z" id="z" value="" size="3">
  <input type="button" name="check" value="Calculate"
     onclick="do_multiply(); return false;">
</body>
</html>
```

6. Performing AJAX Validation

The following *example* validates application for choosing username for login into any chat room group.

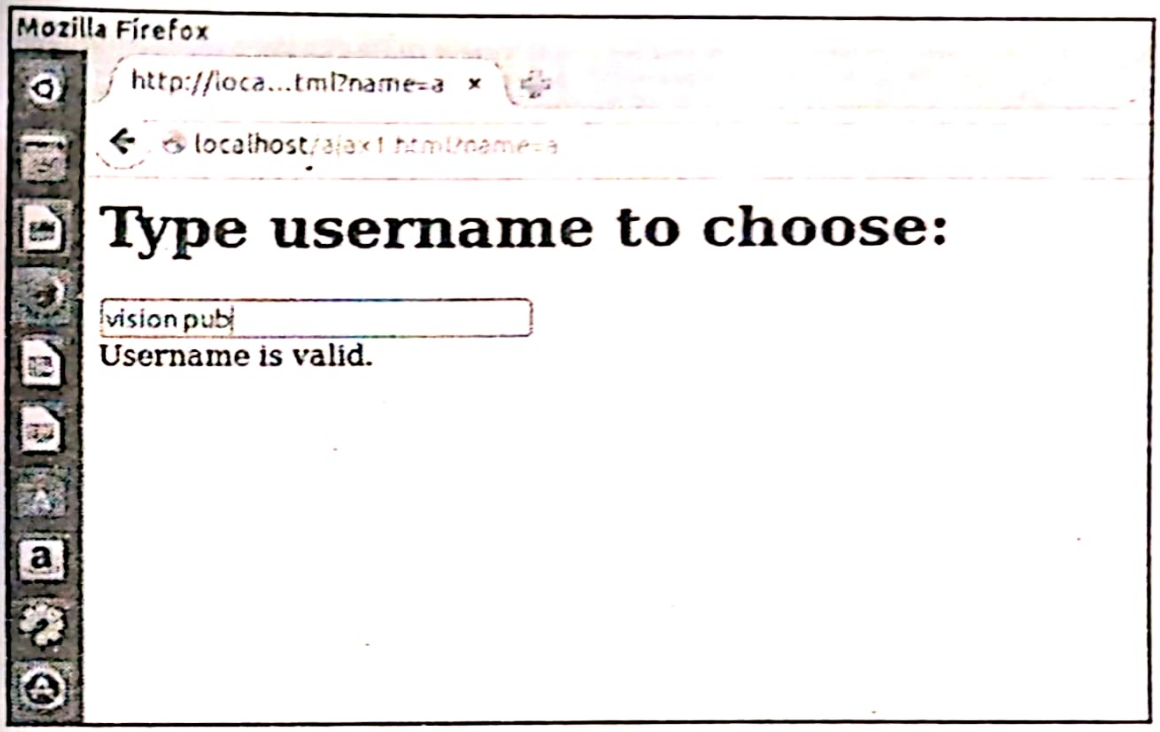
Ajax.html

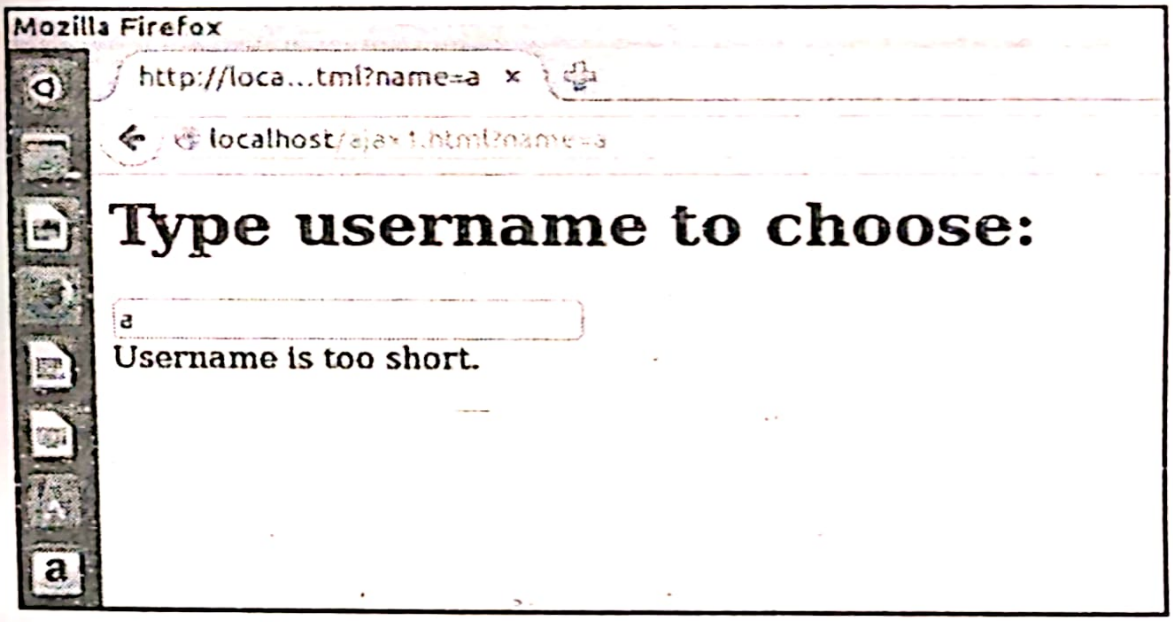
```
<script type="text/javascript">
var http = false;
 if(navigator.appName == "Mozilla")
{
  http = new ActiveXObject("Mozilla.XMLHTTP");
}
else
{
  http = new XMLHttpRequest();
}
function validate(name)
{
  http.abort();
  http.open("GET", "validate.php?name=" + name, true);
```

```
    http.onreadystatechange=function()
    {
      if(http.readyState == 4)
      {
        document.getElementById('res').innerHTML = http.responseText;
      }
    }
    http.send(null);
}
</script>
<h1>Type username to choose:</h1>
<form>
  <input type="text" name="name" onkeyup="validate(this.value)" />
  <div id="res"></div>
</form>
```

Validate.php

```
<?php
function validate($name)
{
  if( $name== '  ')
  {
    return 'Please enter any username';
  }
  if(strlen($name) < 3)
  {
    return "Username is too short.";
  }
  switch($name)
  {
    case 'xyz':
    case 'pqr':
    case 'abc':
    case 'ppp':
    return "Username already exists.";
  }
  return "Username is valid.";
}
echo validate(trim($_REQUEST["name"]));
?>
```





7. Handling XML Data Using PHP and AJAX

AJAX can be used for interactive communication with an XML file. The following *example* demonstrates how a web page can fetch information from an XML file with AJAX.

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 type="text/javascript">
function showCD(str)
{
  if(str=="")
  {
    document.getElementById("txtHint").innerHTML="";
    return;
  }
  if(window.XMLHttpRequest)
  {
    // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
  }
xmlhttp.onreadystatechange=function()
{
  if(xmlhttp.readyState==4 && xmlhttp.status==200)
  {
      document.getElementById("txtHint").innerHTML=xmlhttp.respon
        seText;
    }
  }
  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">Bob</option>
```

```
<option value="Bonnie ">Bonnie</option>
</select>
</form>
<div id="txtHint"><b>CD info will be listed here...</b></div>
</body>
</html>
```

The `showCD()` function does the following:

i. Check if a CD is selected.

ii. Create an XMLHttpRequest object.

iii. Create the function to be executed when the server response is ready.

iv. Send the request off to a file on the server.

v. Notice that a parameter (q) is added to the URL (with the content of the dropdown list).

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

cd_catalog.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<catalog>
<cd>
  <title>emperor the great</title>
  <artist>bob</artist>
  <country>uk</country>
  <company>columbus</company>
  <price>1090</price>
  <year>1950</year>
 </cd>
 <cd>
   <title>search for treasure</title>
   <artist>bonnie </artist>
   <country>russia</country>
   <company> record</company>
   <price>990</price>
    <year>1888</year>
    </cd>
 </catalog>
```

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

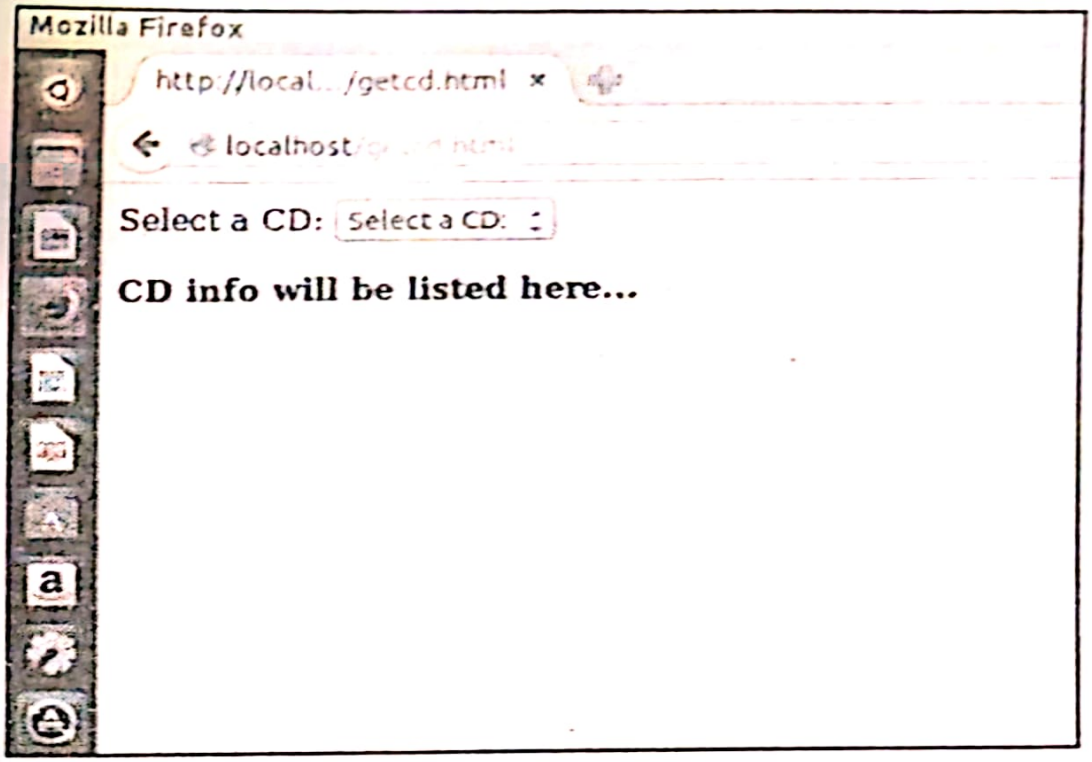
```
<?php
$q=$_GET["q"];
$xmlDoc = new DOMDocument();
$xmlDoc->load("cd_catalog.xml");
$x=$xmlDoc->getElementsByTagName('ARTIST');
for($i=0; $i<=$x->length-1; $i++)
{
  //Process only element nodes
  if($x->item($i)->nodeType==1)
  {
  if($x->item($i)->childNodes->item(0)->nodeValue == $q)
  {
    $y=($x->item($i)->parentNode);
   }
 }
}
$cd=($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 />");
  }
}
?>
```

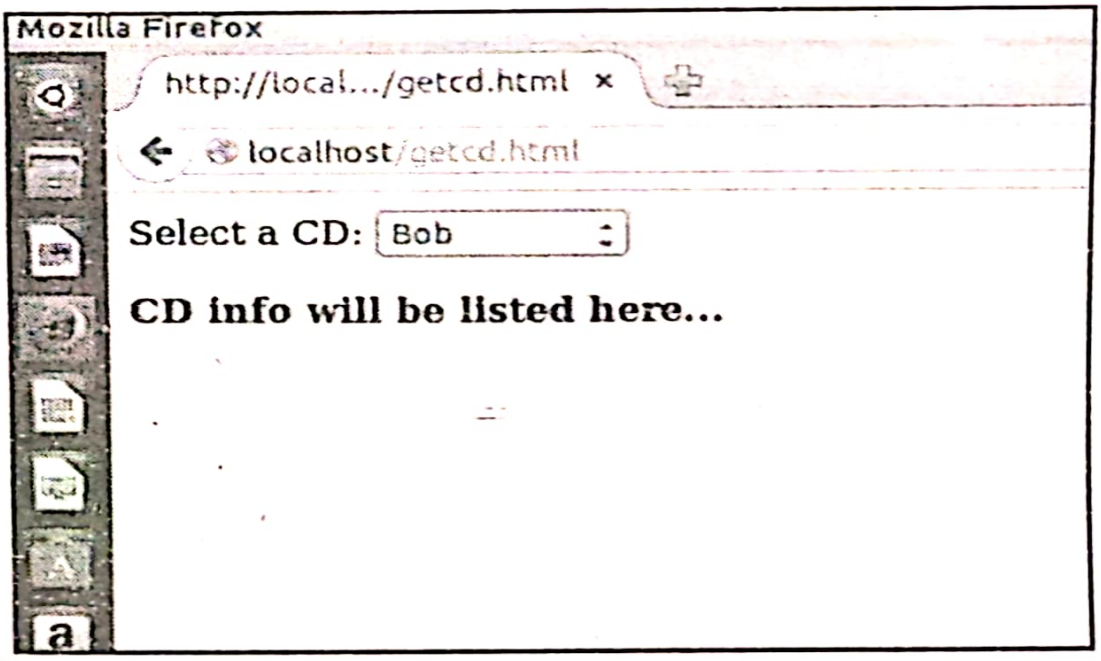
When the CD query is sent from the JavaScript to the PHP page, the following happens:

i. PHP creates an XML DOM object.

ii. Find all <artist> elements that matches the name sent from the JavaScript.

iii. Output the album information (send to the "txtHint" placeholder).





8. XML

XML and HTML are designed with different goals:

i. XML is designed to transport and store data, with focus on what data is.

ii. HTML is designed to display data, with focus on how data looks.

HTML is about displaying information, whereas XML is about carrying information.

XML is the most common tool for data transmission between all sorts of applications, and is becoming more and more popular in the area of storing and describing information.

XML separates data from HTML. With HTML document displaying dynamic data is a bit difficult because, editing HTML each time the data changes is time consuming. With XML, data can be stored in separate XML files.

9. What is XML?

i. XML stands for extensible Markup Language.

ii. XML is a markup language much like HTML.

iii. XML was designed to carry data, not to display data.

iv. XML tags are not predefined. We must define our own tags.

v. XML is designed to be self-descriptive.

vi. XML is a W3C recommendation.

vii. XML simplifies data sharing.

XML data is stored in plain text format. This provides a software and hardware-independent way of storing data.

Features of XML

XML is used in many aspects of web development, often to simplify data storage and sharing.

i. **XML separates data from HTML.**

ii. **XML simplifies data sharing:** In the real world, computer systems and databases contain data in incompatible formats.

XML data is stored in plain text format. This provides a software- and hardware-independent way of storing data which makes it easier to create data that can be shared by different applications.

iii. **XML simplifies data transport:** One of the most time-consuming challenges for developers is to exchange data between incompatible systems over the Internet.

Exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications.

iv. **XML simplifies platform changes:** Upgrading to new systems (hardware or software platforms), is always time consuming. Large amounts of data must be converted and incompatible data is often lost.

XML data is stored in text format which makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.

v. **XML makes data more available:** Different applications can access data, not only in HTML pages, but also from XML data sources.

With XML, data can be available to all kinds of "reading machines" (Handheld computers, voice machines, news feeds, etc.), and make it more available for blind people, or people with other disabilities.

Advantages of XML

i. **XML allows data interchange between computer systems:** XML provides structure for storing data in text format, which can be used as a standard format or protocol for data interchange. Thus, differences in the systems exchanging data become insignificant. Therefore, XML plays an extended role as a very robust data interchange format where as, HTML only decorates the data, it doesn't carry data.

ii. **XML provides domain specific vocabulary:** XML language has no predefined tags: XML allows you to create your own tags based on the requirement of an application, i.e., XML allows to create domain specific vocabulary as per the needs of application.

HTML has predefined tags: HTML documents can only use tags defined in the HTML standard like <b>,<p>,
, etc.

iii. **XML is about carrying information and HTML is about displaying information:** ML was designed to transport and store data, with focus on "what data is". HTML was designed to display data, with focus on "how data looks".

iv. **XML allows granular updates:** In case of HTML, when the data needs to be refreshed, the entire page needs to be refreshed because the page fetches information from the server. This makes the updates slower and causes the entire HTML document to reload.

But, in case of XML, when the data needs to be updated, the entire page need not be reloaded only the changed contents need to be downloaded.

v. **XML enables smart searches:** As HTML provides predefined tags, implementing a search in an HTML document is a difficult task where as, the user defined tags in XML allows smart searches.

Disadvantages of XML

Oct. 2018 – *5M*
What is XML? Give any three advantages and disadvantages of XML.

i. **Lack of application processing:** XML needs an application processing system. There are no browsers yet which can read XML. In case of HTML, anyone can write up a program and that can be read using any browser anywhere in the world. To be able to be read in a browser, XML still depends on HTML, and is not independent of it. The XML documents have to be converted to HTML before they are deployed.

ii. **Repetition:** XML repeats every element and attribute name for every element and attribute instance. In fact, it repeats the element name twice for every instance.

iii. **External references:** The biggest performance risk for XML comes not from the fact that it is text based, that it is parsed, or that it can use Unicode but from the fact that *XML* documents can include external files. To make things worse, the inclusion can

take place in the lowest-level *XML* parsing layer, where it is completely hidden from—and sometimes outside the control of—the application developer.

Uses of XML

The uses of XML are:

i. **Web publishing:** XML allows the developers to save the data into XML files and use the XSLT transformation API's to generate the content in required format such as HTML, XHTML or WML.

ii. **Web searching:** In our application, we can use the XML data and then search the data from the XML file and display to the user. Similar data can be displayed in different format to the user.

iii. **Data transfer:** We can use XML to save configuration or business data for our application. The XML file can be stored, used, transferred to another program.

iv. **Create other languages:** Many languages are created using XML. *Examples* are WML, MathML, etc.

10. XML Document Structure

XML is a **well formed and valid document**. A **well-formed** XML document follows the basic syntax rules and a **valid** document also follows the rules imposed by Document Type Definition (DTD) or an XML Schema.

A **well-formed document** conforms to the XML standard (*for example*, all elements in the document follow the XML specifications).

A **valid** XML document conforms to the DTD associated with the document type, as well as to the XML specifications.

A well-formed XML document may contain:

i. Elements

ii. Attributes

iii. Other constructs allowed by the XML specifications.

10.1 Major Parts of an XML Document

XML documents may contain an optional prolog and then the mandatory root element with an optional section at the end of the data. The following list identifies these major sections:

i. XML documents should contain XML version line and the XML file encoding.

ii. Valid XML documents contain a DTD or an XML schema. A DTD file describes the structure of a document type.

iii. XML documents usually contain one or more elements, each of which may contain one or more attributes.

iv. Elements can contain one or more elements.

v. Attributes contain extra information about a specific tag. XML standards require all attributes to have a value. Values for attributes must be enclosed with single or double quotes.

XML documents may contain additional components such as processing instructions that provide machine instructions for particular applications.

Consider the following document which is well-formed and valid:

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE Client SYSTEM "http://www.examplehtml1/DTD/client.dtd">
<progs>
    <prog>PHP</prog>
    <prog>C++</prog>
</progs>
```

10.2 Well-formed XML Documents

A well formed XML document follows XML specifications following some basic rules. Most common of them are listed below:

1. There is only one parent element which contains the rest of element in the document.

2. *XML declaration:* All XML documents can optionally begin with an XML declaration. The XML declaration provides at a minimum the number of the version of XML in use:

```
<?xml version ="1.0"?>
```

Currently, 1.0 is the only approved version of XML.

The XML declaration can also specify the character encoding used in the document:

```
<?xml version="1.0" encoding="UTF-8"?>
```

All XML parsers are required to support the Unicode "UTF-8" and "UTF-16" encodings; many XML parser support other encodings, such as "ISO-8859-1", as well.

Rules to keep in mind about the XML declaration

i. The XML declaration is case sensitive, it may not begin with "<? XML" or any other variant;

ii. If the XML declaration appears at all, it must be the very first thing in the XML document not even whitespace or comments may appear before it; and

iii. It is legal for a transfer protocol like HTTP to override the encoding value that we put in the XML declaration, so we cannot guarantee that the document will actually use the encoding provided in the XML declaration.

Tags and Elements

XML tags begin with the less-than character ("<") and end with the greater-than character (">"). Tags are used to mark the start and end of **elements**, which are the logical units of information in an XML document.

An element consists of a **start tag**, possibly followed by text and other complete elements, followed by an **end tag**. The following *example* highlights the tags to distinguish them from the text:

```
<p>
  <aname>Peacock</aname> is
  <animal>national animal</animal> of
  <country>India</country>
</p>
```

Note that the end tags include a slash ("/") before the element's name. There are five elements in this *example:*

i. The **p** element, that contains the entire *example* (the **aname** element, the text "is", the **animal** element, the text "of", and the *country* element);

ii. The **aname** element, that contains the text "Peacock";

iii. The **animal** element, that contains the text "national animal";

iv. The **country** element that contains the text "India".

1

Oct. 2018 – *5M*
Explain rules to write XML elements and attributes.

Rules to keep in mind about XML elements

1. **Elements may not overlap**: They must be properly nested. An end tag must always have the same name as the most recent unmatched start tag. The following *example* is not well-formed XML, because "`</aname>`" appears when the most recent unmatched start tag was "`<animal>`":

```
<aname>Peacock<animal>national animal</aname> </animal>
```

The following *example* shows the tags properly nested:

```
<aname>Peacock</aname><animal>national animal</animal>
```

2. An XML document has **exactly one root element.** As a result, the following *example* is not a well-formed XML document, because both the *a* and *b* elements occur at the top level which is incorrect:

```
<!-- WRONG! -->
<a>...</a>
<b>...</b>
```

Which can be rectified by including a and b elements within a new *x* root element:

```
<x>
<a>...</a>
<b>...</b>
</x>
```

3. XML element (and attribute) names are case-sensitive, so "aname" and "Aname" refer to different elements.

4. XML has a special **empty-element tag** that represents both the start tag and the end tag:

```
<p>some data<hr/>
For output</p>
```

In this *example*, "`<hr/>`" represents both the start and the end of the *hr* element; it could have been written as "`<hr></hr>`" (which is exactly equivalent).

XML Elements and Attributes

XML documents form a tree structure that starts at "the root" and branches to "the leaves".

XML documents must contain a **root element**. This element is "the parent" of all other elements.

The elements in an XML document form a document tree. The tree starts at the root and branches to the lowest level of the tree.

All elements can have sub elements (child elements):

```
<root>
<child>
    <subchild>.....</subchild>
  </child>
</root>
```

The terms parent, child are used to describe the relationships between elements. Parent elements have children. Children on the same level are called siblings (brothers or sisters).

All elements can have text content and attributes (just like in HTML).

Example,

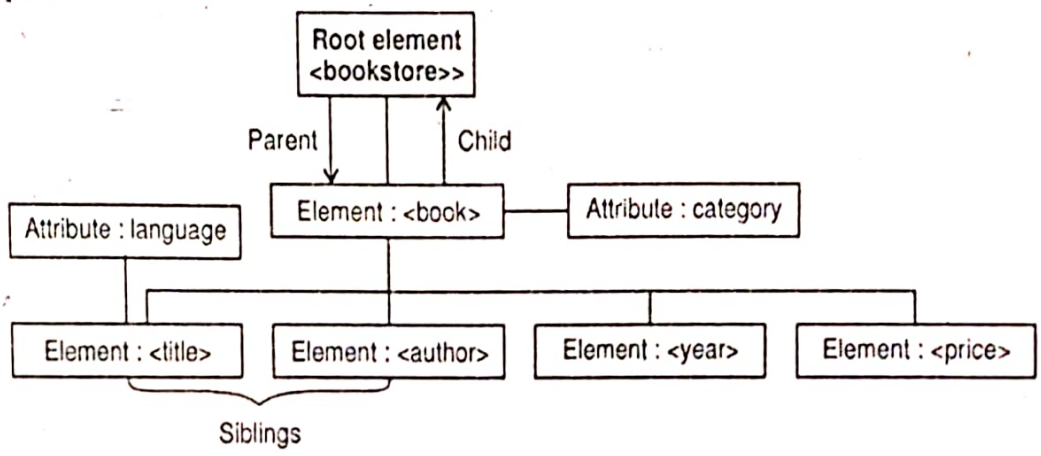


Figure 6.6

```
<?xml version="1.0" encoding="UTF-8" ?>
<bookstore>
<book category="SCIENCE">
        <title lang="en">DiscoverIndia</title>
        <author>Gayatridevi</author>
        <year>2001</year>
        <price>3000</price>
</book>
<book category="TravelLiving">
        <title lang="en">Asiancultures</title>
        <author>Satyajitray</author>
        <year>2005</year>
        <price>2999</price>
</book>
</bookstore>
```

The root element in the *example* is `<bookstore>`. All `<book>` elements in the document are contained within `<bookstore>`.

The `<book>` element has 4 children: `<title>, <author>, <year>, <price>`.

Rules to remember about XML attribute

1. Attribute names in XML (unlike HTML) are case sensitive: HREF and href refer to two different XML attributes.
2. We must not provide two values for the same attribute in the same start tag. The following *example* is not well-formed because the b attribute is specified twice:

   ```
   <a b="x" c="y" b="z">....</a>
   ```

3. Attribute names should never appear in quotation marks, but attribute values must always appear in quotation marks in XML (unlike HTML) using the "or" characters. The following *example* is not well-formed because there are no delimiters around the value of the b attributes.

   ```
   <!-- WRONG! -->
   <a b=x>...</a>
   ```

References

A **reference** allows us to include additional text or markup in an XML document. References always begin with the character "&" (which is specially reserved) and end with the character ";".

XML has two kinds of references

1. **Entity references:** An entity reference, like "&", contains a name (in this case, "amp") between the start and end delimiters. The name refers to a predefined string of text and/or markup, like a macro in the C or C++ programming languages.
2. **Character references:** A character reference, like "&", contains a hash mark ("#") followed by a number. The number always refers to the Unicode for a single character, such as 65 for the letter "A".

XML also provides five pre-declared entities that we can use to escape special characters in an XML document:

Character	Predeclared Entity
&	&amp;
<	&lt;
>	&gt;
"	&quot;
'	&apos;

For example, if we want to display the name "`smith&smith`" should appear in the XML markup as "`smith&smith`": the XML parser will take care of changing "&" back to "&" automatically when the document is processed.

XML Data Binding

XML data binding refers to a means of representing information in an XML document as a business object in computer memory. This allows applications to access the data in the XML from the object rather than using the DOM or SAX to retrieve the data from a direct representation of the XML itself.

11. PHP and XML

XML and PHP can be combined to build Web applications. It includes detailed explanations of PHP's XML extensions, together with illustrations of using PHP to parse, validate and transform XML mark-up, traverse XML data trees, exchange data between Web applications, overlay remote procedure calls over HTTP and use free open-source tools to add new capabilities to our XML/PHP applications.

XML and PHP serves both as an implementation guide to the topic and as a handy desktop reference for quick lookups, combining all the features required.

The most common XML functions are :

`xml_parser_create(), xml_parse_into_struct(), xml_get_error_code().`

Example

demoxml.html

```
<html>
<body>
<form action="demoxml.php" method="POST">
   root element <input type="text" name="start">
   element name <input type="text" name="ename">
   attribute name <input type="text" name="evalue">
<input type="submit" name="cmdbutton" value="create">
<input type="hidden" name="hidden" value="true">
</form>
</body>
</html>
```

demoxml.php

```
<?php
if(isset($_POST['hidden']))
{
   echo "Links Data Posted";
         /* Data from the form is now being stored in variables
              in string format */
        $start = $_POST['start'];
        $ename = $_POST['ename'];
        $evalue = $_POST['evalue'];
        $xmlBeg = "<?xml version='1.0' encoding='ISO-8859-1'?>";
        $rootElementStart = "<$start>";
        $rootElementEnd = "</$start>";
        $xml_document = $xmlBeg;
        $xml_document.= $rootELementStart;
```

```
        $xml_document.= "<fname>";
        $xml_document.= $ename;
        $xml_document.= "</fname>";
        $xml_document.= "<sname>";
        $xml_document.= $evalue;
        $xml_document.= "</sname>";
        $xml_document.= $evalue;
        $xml_document.= $rootElementEnd;
        $path_dir.= $start.".xml";
    /* Data in Variables ready to be written to an XML file */
        $fp = fopen($path_dir,'w');
        $write = fwrite($fp,$xml_document);
        echo "<br>  THE LOaded file is <br>" .$path_dir. "<br>";
}
?>
```

Input

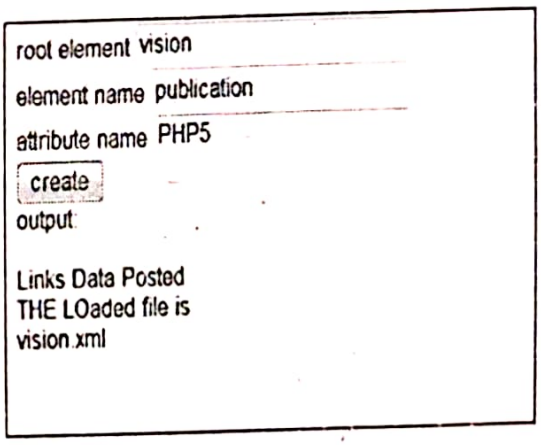


Figure 6.7

Output

Links Data Posted

THE LOaded file is

vision.xml

Open the file vision.xml in the editor:

```
#vi vision.xml
<?xml version='1.0' encoding='ISO-8859-1'?>
<vision>
<fname>publication</fname>
<sname>PHP5</sname>
</vision>
```

12. XML Parser

Most browsers have a built-in XML parser to read and manipulate XML.

The parser converts XML into a JavaScript accessible object (the XML DOM).

The XML DOM contains methods (functions) to traverse XML trees, access, insert and delete nodes.

However, before an XML document can be accessed and manipulated, it must be loaded into an XML DOM object. Most browsers have a built-in XML parser.

1. **xml_parser_create():** Function creates an XML parser. This function returns a resource handle to be used by other XML functions on success, or FALSE on failure.

   Syntax

   ```
   xml_parser_create(encoding);
   ```

   The parameter encoding is optional and it specifies the output encoding.

2. **xml_parse_into_struct():** Function parses XML data into an array. This function parses the XML data into 2 arrays:

   i. Value array: Containing the data from the parsed XML.

   ii. Index array: Containing pointers to the location of the values in the Value array.

   This function returns 1 on success, or 0 on failure.

   Syntax

   ```
   xml_parse_into_struct(parser,xml,value_arr,index_arr);
   ```

Parameter	Description
parser	Required. Specifies XML parser to use.
xml	Required. Specifies XML data to parse.
value_arr	Required. Specifies the target array for the XML data.
index_arr	Optional. Specifies the target array for index data.

The following *example* demonstrates the use of `xml_parser_create()` and `xml_parse_into_struct()` functions.

The data.xml file

```
<?xml version="1.0"?>
<user>
    <firstname>xyz</firstname>
    <surname>pqr</surname>
    <address>100</address>
    <contact>
      <phone>4444</phone>
      <url>http://hello.org</url>
      <email>xyz@pqr.com</email>
    </contact>
 </user>
```

PHP code

```
<?php
//invalid xml file
$xmlfile = 'data.xml';
$xmlparser = xml_parser_create();
// open a file and read data
$fp = fopen($xmlfile, 'r');
$xmldata = fread($fp, 4096);
xml_parse_into_struct($xmlparser,$xmldata,$values);
xml_parser_free($xmlparser);
print_r($values);
?>
```

13. The XML DOM (XML Document Object Model)

The XML DOM (XML Document Object Model) defines a standard way for accessing and manipulating XML documents. The DOM extension follows the Worldwide Web Consortium's DOM Level 2 recommendation which states "The DOM Application Programming Interface(API) for valid HTML and well formed XML documents".

Configuring PHP with-dom = dom_dir makes this extension available.

PHP DOMs Functions

i. **domxml_new_doc():** To create new XML document.

ii. **domxml_open_file():** To open an XML document file as a DOM object.

iii. **domxml_open_mem():** Create a DOM object from an XML document already in memory.

iv. **create_element():** Creates a new element.

v. **append_child():** Which appends an element as a child of an existing element.

vi. **DomDocument->get_elements_by_tagname:** Returns array with nodes with given tagname in document or empty array, if not found.

Example,

So if we have a document:

```
<root>
  <a>
    <b>
      <c>
      </c>
    </b>
  </a>
</root>
```

Calling

```
$a->get_elements_by_tagname("b")
```

returns an array containing the element <b>, as expected.

For example, to open an XML file the DOM object is created as:

```
$my_dom_obj.
$my_dom_obj=doxml_open_file("firstxml.xml");
```

To create a variable representing the root element found within the document

```
$root_elem=$dom->document_element();
```

Consider the *following example* which demonstrates this:

Example 1

```
<?php
$xml_file="firstxml.xml";
$element = new DomElement();
if(!$dom = domxml_open_file($xml_file))
{
  die("Error opening xml file");
}
$tables = $dom->get_elements_by_tagname("table");
foreach($tables as $table)
{
  $element = $table;
  echo "Attribute name: ", $element->get_attribute("name");
}
?>
```

Example 2

```
<?xml version="1.0" ?>
<books>
<book>
        <author>XYZ</author>
        <title>PHP</title>
        <publisher>Vision</publisher>
</book>
<book>
        <author>PQR</author>
        <title>PHP</title>
        <publisher>Vision</publisher>
</book>
</books>
<?php
// DOMElement->getElementsByTagName() -- Gets elements by tagname
// nodeValue: The value of this node, depending on its type.
// Load XML File. we can use loadXML
        we wish to load XML data from a string
$doc = new DOMDocument();
$doc->load('books.xml');
  $books = $doc->getElementsByTagName("book");
// for each note tag, parse the document and get values for
```

```
  // tasks and details tag.
foreach($books as $book)
{
  $authors = $book->getElementsByTagName("author");
  $author = $authors->item(0)->nodeValue;
  $publishers = $book->getElementsByTagName("publisher");
  $publisher = $publishers->item(0)->nodeValue;
  $titles = $book->getElementsByTagName("title");
  $title = $titles->item(0)->nodeValue;
  echo "$title - $author - $publisher\n";
}
?>
```

We can run the PHP script on the command line like this:

```
# php e1.php
PHP - XYZ - Vision
PHP - PQR - Vision
#
```

Example 3

The following *example* demonstrates how to insert multiple nodes into an existing XML document.

```
<?php
$dom=new DOMDocument('1.0','iso-8859-1');
$rootElement=$dom->createElement('rootnode','');
$secondElement=$dom->createElement('secondnode','Second element:new DOM
document');
$thirdElement=$dom->createElement('thirdnode','Third element: new
  DOM document');
// insert the new elements into the document
$dom->appendChild($rootElement);
$rootElement->appendChild($secondElement);
$rootElement->appendChild($thirdElement);
// tell browser the output is XML via the 'Content-Type' HTTP
// header
header('Content-Type: text/xml');
// display DOM document
echo $dom->saveXML();
?>
```

14. SimpleXML

SimpleXML is an extension of PHP. This extension has a set of functions that are used to handle the XML and to convert them into an object. After the conversion to an object it is easy to process the XML as we do with other conventional objects.

SimpleXML converts the XML document into an object, like this:

1. **Elements:** Are converted to single attributes of the SimpleXML element object. When there's more than one element on one level, they're placed inside an array.
2. **Attributes:** Are accessed using associative arrays, where an index corresponds to the attribute name.
3. **Element data:** Text data from elements are converted to strings. If an element has more than one text node, they will be arranged in the order they are found.

SimpleXML is fast and easy to use when performing basic tasks like:

i. Reading XML files
ii. Extracting data from XML strings
iii. Editing text nodes or attributes

14.1 SimpleXML Extension

SimpleXML includes functions for working with XML documents that make common operations fairly easy.

The four simple rules are:

1. Properties denote element iterators.
2. Numeric indices denote elements.
3. Non-numeric indices denote attributes.
4. String conversion allows access to TEXT data.

By using these four rules, we can access all the data from the XML file.

1. **Properties denote element iterators:** Which means that we can loop over all <p> tags in the <body>, like this:

```
<?php
foreach ($sx2->body->p as $p)
{
}
?>
```

2. **Numeric indices denote elements:** Which means that we can access the second <p> tag with

```
<?php
   $sx->body->p[1];
?>
```

3. **Non-numeric indices denote attributes:** Which means that we can access the background attribute of the body tag with

```
<?php
   echo $sx->body['background'];
?>
```

4. **String conversion allows access to TEXT data:** Means we can access all text data from the elements. The following code, echos the contents of the second <p> tag (thus combining rules 2 and 4):

```
<?php
   echo $sx->body->p[1];
?>
```

Accessing TEXT data from a node will not include its child nodes.

The asXML() method includes child nodes but also adds all the text, strip_tags() method prevents this.

The following example demonstrates this:

```
<?php
   echo strip_tags($sx->body->p[1]->asXML()). "\n";
?>
```

If we want to iterate over all child elements of the body node, use the `children()` method of the SimpleXML element object. The following example iterates over all children of `<body>`.

```
<?php
   foreach($sx->body->children() as $element)
   {
      /* do something with the element */
   }
?>
```

If we want to iterate over all the attributes of an element, the `attributes()` method is available. Let's iterate over all the attributes of the first `<a>` tag.

```
<?php
foreach($sx->body->p[0]->a->attributes() as $attribute)
{
   echo $attribute . "\n";
}
?>
```

The SimpleXML functions include:

1. **simplexml_load_file:** This function will convert the well-formed XML document in the file specified by filename to an object of class SimpleXMLElement.
2. **simplexml_load_string:** This function will take the well-formed xml string data and return an object of class SimpleXMLElement with properties containing the data held within the xml document.
3. **simplexml_import_dom:** This function takes a node of a DOM document and makes it into a SimpleXML node.
4. **simplexmlElement->xpath:** The xpath method searches the SimpleXML node for children matching the xpath path. It always returns an array of SimpleXMLElement objects.
5. **simplexml_element->children:** This method finds the children of the element of which it is a member.
6. **simplexml_element->attributes:** This function provides the attributes and values defined within an xml tag.
7. **simplexmlElement->asxml:** The asXML method formats the parent object's data in XML version 1.0.

Creating a SimpleXML Object

We can create a SimpleXML object as shown in this *example*.

The data.xml file:

```
<?xml version="1.0" ?>
 <user>
    <firstname>xyz</firstname>
    <surname>pqr</surname>
    <address>100</address>
    <contact>
      <phone>4444</phone>
      <url>http://hello.org</url>
      <email>xyz@pqr.com</email>
      </contact>
  </user>
<?php
if(file_exists('data.xml'))
{
   $xml = simplexml_load_file('data.xml');
   var_dump($xml);
}
else
{
    exit('Error.');
}
?>
```

Note SimpleXML extension and the DOM extension has related functions which allow us to share the same XML structure between both extensions

```
<?php
/*** create a SimpleXML object ***/
if(!$xml = simplexml_load_file('data.xml'))
{
   echo "Unable to load XML file";
}
else
  {
     echo $xml->user[1]->firstname.' '. $xml->user[1]-
               >surname.'<br />';
     echo $xml->user[1]->contact->phone;
  }
?>
```

Accessing the attributes

```
<?php
   /*** create a SimpleXML object ***/
   if(!$xml = simplexml_load_file("data.xml"))
   {
     echo "Unable to load XML file";
   }
  else
  {
    $i = 0;
   /*** loop over the elements ***/
   foreach($xml as $node)
   {
     echo $xml->user[$i]->contact->phone->attributes().'<br />';
        $i++;
   }
}
?>
```

The following *example* uses `simplexml_load_string()`

```
<? php
$string = <<<XML
<?xml version='1.0'?>
<document>
  <cmd>login</cmd>
  <login>Richard</login>
</document>
XML;
$xml = simplexml_load_string($string);
print_r($xml);
$login = $xml->login;
print_r($login);
$login = (string) $xml->login;
print_r($login);
?>
```

Browsing SimpleXML Objects

The **SimpleXML extension**, enabled by default in PHP 5, is the easiest way to work with XML. We have to simply access the XML through a data structure representation.

Storing SimpleXML Objects

We can store a changed or manipulated structure or a subnode to disk. We use the `asXML()` method to do this, which we can call on any SimpleXML object.

```
<?php
    file_put_contents('filename.xml', $sx2->asXML());
?>
```

15. Changing a Value with SimpleXML

XML document parts cannot only be read into SimpleXML objects but also change data in memory document.

Changing the value of a node:

```
<?php
  $xmlstr= <<< XML
<phpprog>
<pname name= "C++">
<platform>windows</platform>
</pname>
<pname name="PHP">
<platform>Linux</platform>
</pname>
</phpprog>
  XML;
  $dom=new domDocument;
  $dom->loadXML($xmlstr);
  $fstring = simplexml_import_dom($dom);
  $fstring_xml->program[0]->platform= "Ubuntu";
  echo "<PRE>";
  var_dump($fstring_xml);
  echo"</PRE>";
?>
```

Example to create an XML document by accepting the values from HTML page:

test.html

```
<html>
<head>
     <title>XML Links Data</title>
</head>
<body>
     <p><center>Links Data</center></p>
<form method="POST" action="xml.php">
<input type="hidden" name="hidden" value="true">
<table>
<tr> <td>URL:<input type="text" name="urlAdd" size="50"></td> </tr>
<tr><td>Url Description:<input type="text" name="urlDes"
  size="50"></td></tr>
  <tr><td>Document Name:<input type="text" name="urlDoc"
  size="50"></td> </tr>
</table>
<p align="center">
<input type="submit" value="Submit" name="create"></p>
</form>
</body>
 </html>
```

xml.php

```
<?php
if(isset($_POST['hidden']))
{
   echo "Links Data Posted";
   /* Data from the form is now being stored in variables in
   string format */
   $urlDoc = $_POST['urlDoc'];
   $urlAdd = $_POST['urlAdd'];
   $urlDes = $_POST['urlDes'];
   $xmlBeg = "<?xml version='1.0' encoding='ISO-8859-1'?>";
   $rootELementStart = "<$urlDoc>";
   $rootElementEnd = "</$urlDoc>";
   $xml_document= $xmlBeg;
   $xml_document.= $rootElementStart;
   $xml_document.= "<site>";
   $xml_document.= $urlAdd;
   $xml_document.= "</site>";
```

```
    $xml_document.= "<description>";
    $xml_document.= $urlDes;
    $xml_document.= "</description>";
    $xml_document.= $rootElementEnd;
    $path_dir.= $urlDoc.".xml";
    /* Data in Variables ready to be written to an XML file */
    $fp = fopen($path_dir,'w');
    $write = fwrite($fp,$xml_document);
    /* Loading the created XML file to check contents */
    $sites = simplexml_load_file("$path_dir");
    echo "<br> Checking the loaded file <br>" .$path_dir. "<br>";
    echo "<br><br>Whats inside loaded XML file?<br>";
    print_r($sites);
}
?>
```

Import a DOM with Simple XML

```
<?php
$xmlstr= <<< XML
<phpprog>
   <pname name = "VisualBasic">
   <platform>windows</platform>
</pname>
   <pname name="PHP">
<platform>Linux</platform>
</pname>
</phpprog>
XML;
   $dom = new domDocument;
   $dom->loadXML($xmlstr);
   $s=simplexml_import_dom($dom);
   echo "the platform for the first program is <B>". $s->pname[0]->
        platform. "</B>";
?>
```

Some more *examples* of simplexml.

The following script creates a parent user element and then a firstname and surname child elements.

```
<?php
try
{
    /*** a new dom object ***/
```

```
    $dom = new domDocument;
    /*** make the output tidy ***/
    $dom->formatOutput = true;
    /*** create the root element ***/
    $root = $dom->appendChild($dom->createElement("users"));
    /*** create the simple xml element ***/
    $sxe = simplexml_import_dom($dom);
        /*** add a user node ***/
    $user = $sxe->addchild("user");
    /*** add a firstname element ***/
    $user->addChild("firstname", "John");
    /*** add a surname element ***/
      $user->addChild("surname", "Brady");
    /*** add address element ***/
    $user->addChild("address", "1 Bunch St");
    /*** add the city element ***/
    $user->addChild("city", "Downtown");
    /*** add the country ***/
    $user->addChild("country", "America");
    $contact = $user->addChild("contact");
    $contact->addChild("phone", "4444 4444");
    $contact->addChild("url", "http://phpro.org");
    $contact->addChild("email", "brady@bunch.example.com");
    echo $sxe->asXML();
}
catch(Exception $e)
{
   echo $e->getMessage();
}
?>
```

Difference between DTDs and Schema

DTD, or Document Type Definition, and XML Schema, which is also known as XSD, are two ways of describing the structure and content of an XML document.

The first difference between DTD and XML Schema, is namespace awareness; XML Schema is, while DTD is not. Namespace awareness removes the ambiguity that can result in having certain elements and attributes from multiple XML vocabularies, by giving them namespaces that put the element or attribute into context.

Another key advantage of XML Schema, is its ability to implement strong typing. An XML Schema can define the data type of certain elements, and even constrain it to within specific lengths or values. This ability ensures that the data stored in the XML document is accurate. DTD lacks strong typing capabilities, and has no way of validating the content to data types.

Solved Programs

1. Write PHP script to read emp.xml file (contains emp_no, emp_name, salary, designation) and print employee details in tabular format (use simple XML).

Solution

emp.xml

```
<?xml version='1.0'?>
<employees>
    <emp>
        <empno>1111</empno>
        <name>Vision  Wilson</name>
        <sal>50000</sal>
        <desig>Manager</desig>
    </emp>
    <emp>
         <empno>2222</empno>
         <name>Shiney Wilson</name>
         <sal>50000</sal>
         <desig>Lecturer</desig>
    </emp>
</employees>
```

Emp.php

```
<html>
<body>
<table border=1>
<?php
 // load XML file
$xml = simplexml_load_file('empl.xml') or die ("Unable to load XML");
// access xml element
echo '<tr><th>empno</th>
<th>name</th>
<th>sal</th>
<th>desig</th>   </tr>';
 foreach($xml as $emp)
```

```
  {
    echo'<tr><td>'.$emp->empno.'</td>';
    echo'<td>'.$emp->name.'</td>';
    echo'<td>'.$emp->sal.'</td>';
    echo'<td>'.$emp->desig.'</td></tr>';
  }
?>
</table>
</body>
</html>
```

2. Write a PHP script to read book.xml and print book details in tabular format using simple XML. (Contents of book.XML are (bookcode, bookname, author, publisher, price).

Solution

book.xml

```
<?xml version="1.0" ?>
<books>
 <book>
   <bookcode>computer</bookcode>
   <bname>PHP</bname>
   <author>vision</author>
    <pub>Vision Publications</pub>
      <price>Rs.120</price>
      </book>
</books>
```

book.php

```
<?php
    if(! $xml =simplexml_load_file("book.xml"))
    {
      echo "Unable to load XML file";
    }
    else
    {
       echo $xml->getName() . "<br />";
       echo"<tableborder='1'><tr><th>BookCode</th><th>BookName</th>
       <th>Author</th>
       <th>Publisher</th><th>Price</th></tr>";
```

```
        foreach($xml->children() as $child)
        {
         echo "<tr>";
         echo $child->getName() . ": " . $child . "<br />";
         foreach($child->children() as $innerchild)
            {    echo "<td>".$innerchild."</td>";
            }
   echo "</tr>";
}
echo "</table>";
    }
?>
```

3. Write a php script to read item-XML file(contain INo, Iname, I-desc,Price) and print item details in tabular format(using simple XML).

Solution

item.xml

```
<? xml version="1.0"?>
<Items>
  <Item>
       <Ino>1</Ino>
       <Iname>Dove</Iname>
       <Idesc>Shampoo</Idesc>
       <Price>100</Price>
   </Item>
<Item>
     <Ino>2</Ino>
     <Iname>Pears</Iname>
     <Idesc>FaceWash</Idesc>
     <Price>50</Price>
     <Item>
</Items>2
```

item1.php

```
<?php
$xml=simplexml_load_file('item.xml') or die("Cannot load");
echo "<table border=1>";
echo "<tr><th>Ino</th><th>Iname</th><th>I-
desc</th><th>Price</th></tr>";
foreach($xml->Item as $a)
{
   echo "<tr><td>".$a->Ino."</td><td>".
   $a->Iname."</td><td>".$a->Idesc."</td><td>".
```

```
    $a->Price."</td></tr>";
}
?>
```

4. **Write a php script to read Film.XML file which contains film number, name of film, name of Director, name of Producer, release year, Actor name, budget. Print film details of specific actor in tabular format after accepting name of Actor as input.**

Solution

XML file containing the details of various films (Film.xml)

```
<films>
  <film>
    <filmnumber>1</filmnumber>
    <nameoffilm>xxx</nameoffilm>
    <nameofdirector>yyy</nameofdirector>
    <nameofproducer>zzz</nameofproducer>
    <releaseyear>2000</releaseyear>
    <actorname>aaa</actorname>
    <budget>400000</budget>
</film>
<film>
  <filmnumber>2</filmnumber>
  <nameoffilm>aaa</nameoffilm>
  <nameofdirector>bbb</nameofdirector>
  <nameofproducer>ccc</nameofproducer>
  <releaseyear>2010</releaseyear>
  <actorname>aaa</actorname>
  <budget>100000</budget>
</film>
  <film>
    <filmnumber>3</filmnumber>
    <nameoffilm>abc</nameoffilm>
    <nameofdirector>def</nameofdirector>
    <nameofproducer>ghi</nameofproducer>
    <releaseyear>2013</releaseyear>
    <actorname>aaa</actorname>
    <budget>300000</budget>
  </film>
</films>
```

Program to accept actor name from the user (Film.html)

```
<HTML>
  <form action="Film.php" method="post">
      Enter actor name<input type="text" name="t1">
    <input type=submit value="send">
</form>
</HTML>
```

Program to print film details of specific actor in tabular format based on the Actor name (Film.php)

```
<HTML>
<H1> Film details of specific actor </H1>
  <TABLE border="1" >
    <TR><TH>Film number</TH>
    <TH>Name of film</TH>
    <TH>Name of Director</TH>
  <TH>Name of Producer</TH>
  <TH>Release year</TH>
  <TH>Actor name</TH>
  <TH>Budget</TH>
</TR>
<?php
  $aname=$_POST['txtaname'];
  $finfo = simplexml_load_file("File.xml") or die("Cannot load
File.xml");
  foreach($finfo->Films as $f)
  {
    if($f->actorname== $aname)
    {
      echo "<TR>";
      echo"<TD>$f->filmnumber</TD>\
          <TD>$f->nameoffilm</TD>
          <TD>$f>nameofdirector</TD>
          <TD>nameofproducer</TD><TD>releaseyear</TD>
          <TD>actorname</TD><TD>budget</TD>";
      echo "</TR>";
} }
?>
</TABLE>
</BODY>
</HTML>
```

5. Write a PHP script to read project.xml which contains (project id, name, start date, deadline) and print it in tabular format.

Solution

```
<?php
$xml=simplexml_load_file('project.xml') or die("cannot load
project.xml");
echo"<table border = '1'>";
echo"<tr><th>Project id</th><th>Name</th><th>Start
date</th><td>Deadline</th></tr>";
foreach($xml->project as $p)
{
  echo "<tr><td>".$p['projectid']."</td>";
  echo "<tr><td>".$p['name']."</td>";
  echo "<tr><td>".$p['startdate']."</td>";
  echo "<tr><td>".$p['deadline']."</td></tr>";
}
echo"</table>";
?>
```

6. Write a PHP script to read Account.XML file which contains Account-No, name, address, branch, Account type, balance. Print Account details of specific branch in tabular format after accepting branch as input.

Solution

```
<HTML>
<HEAD>
<TITLE>Account Details</TITLE>
</HEAD>
<BODY>
<H1> Account Details</H1>
<?php
$f = simplexml_load_file("ACCOUNT.xml") or die("Cannot load
ACCOUNT.xml");
function generateForm()
{
  ?>
  <FORM action="<?php echo $_SERVER['PHP_SELF'] ?>" method="GET">
  Enter Branch name: <INPUT type="text" name="branch_name"><BR><BR>
  <INPUT type="submit" name="submit" value="Go"/><BR></FORM>
  <?php
}
$brname = $_GET['branch_name'];
if(strlen(trim($brname)) == 0)
```

```
{
  echo "Error: Branch Name field is blank.";
  break;
}
echo "Account Details of specific $brname are: <BR>";
echo "<TABLE border=\"1\" cellpadding=\"5\">";
echo"<TR><TH>Account-
no</TH><TH>Name</TH><TH>Address</TH><TH>Branch</TH><TH>Account
Type</TH><TH>Balance</TH></TR>";
foreach($f->Account as $account)
{
  if($account->branch == $brname)
  {
     echo "<TR><TD>$account->Account_no</TD><TD>$account-
         >name</TD><TD>$account->address</TD><TD>$account-
           >branch</TD><TD>$account->accounttype</TD><TD>$account-
             >balance</TD></TR>";
} }
echo "</TABLE>";
break;
generateForm();
?>
</BODY>
</HTML>
```

SUMMARY

- AJAX stands for Asynchronous JavaScript and XML.
- An Ajax application eliminates the start-stop-start-stop nature of interaction on the Web by introducing an intermediary an Ajax engine between the user and the server.
- AJAX can be used for interactive communication with an XML file.
- Ajax applications can be connected with the databases using php programs.
- XML was basically designed to transport and store data whereas HTML was designed to display data.
- XML stands for eXtensible Markup Language.
- XML is a well formed and valid document.
- XML documents should contain XML version line.
- XML documents usually contain one or more elements, each of which may contain one or more attributes.
- XML elements can be made from start < > and end < / > tags or can be single tag with a terminator< / >.
- XML elements must be properly nested.
- XML documents must contain a root element.
- XML and PHP can be combined to build cutting-edge Web applications.
- The XML DOM (XML Document Object Model) defines a standard way for accessing and manipulating XML documents.
- SimpleXML includes functions for working with XML documents that make common operations fairly easy.

Exercises

A. Choose the correct option: [1 Mark]

1. AJAX is based on
   a. JAVASCRIPT AND xml
   b. Vbscript and xml
   c. Javascript and java
   d. Javascript and http request

2. The xmlhttprequest object supports a method called "quit".
   a. True
   b. False

3. When a user views a page containing a javascript program, which machine actually executes the script?
   a. The Web server
   b. The User's machine running a Web browser
   c. A central machine deep within Netscape's corporate offices
   d. None of the above

4. What combination of technologies gives AJAX its name?
   a. Atlas and XML
   b. Autonomic Computing and DHTML
   c. Asynchronous javascript and XML
   d. ASP and XAML

5. What makes Ajax unique?
   a. It makes data requests asynchronously.
   b. It uses C++ as its programming language.
   c. It works the same with all Web browsers.
   d. It works as a stand-alone Web-development tool.

6. XML stands for
   a. Extra Markup Language
   b. Explore Markup Language
   c. Expensive Markup Language
   d. Extensible Markup Language

7. Main purpose to design XML is
   a. Save and transport data
   b. How to reuse data
   c. To made connection b/w data
   d. How to use data

8. Which one is correct?
   a. <NAME><hello>visions</hello></NAME>
   b. <name><hello>visions</name></hello>
   c. <NAME><HELLO>visions</hello></name>
   d. None of the above

B. Answer the following questions: **[1 Mark]**

1. Give any two applications of AJAX.
2. Which framework can be used for dynamic data filtering?
3. What is an AJAX?
4. Write any two applications of XML.
5. Give names of XML Parser.
6. Write function to load an XML file into an object.
7. What is DOM?
8. Write any two applications of XML.
9. Explain rules to write xml elements and attributes.
10. Give names of two XML parser.
11. Write any two applications of XML.
12. What is DomDocument( )?
13. Explain which functions are used to identify the browser.
14. "XML is case sensitive". Justify T/F.
15. What is XML Parser?
16. What is Binding?

C. Answer the following questions: **[5 Marks]**

1. Write a note on AJAX php frame work.

2. Write an AJAX program to display list of games stored in an array on clicking ok button.
3. Write advantages and disadvantages of AJAX.
4. Write an AJAX script to search employee name from employee.dat file.
5. Write a simple php program which implements Ajax for addition of two numbers.
6. Write a short note on role of ajax engine in synchronization of Ajax programs.
7. Employee(id, name, address, designation, salary)

   Write an ajax program to accept name and salary of employee and increase employee salary by 10% in the database.
8. Write an Ajax program to print the content of the file myfile.dat.
9. Write a note on Ajax web application model.
10. Write an Ajax program to search student name according to the characters typed and display list using array.
11. What is DOM? How does it relate to XML?
12. Write a PHP script to read Account.XML file which contains Account-No, name, address, branch, Account type, balance. Print Account details of specific branch in tabular format after accepting branch as input.
13. Give advantages of XML over HTML.
14. Write a PHP script to read project.xml which contains (project id, name, start date, deadline) and print it in tabular format.
15. Write a php script to read Film.XML file which contains film number, name of film, name of Director, name of Producer, release year, Actor name, budget. Print film details of specific actor in tabular format after accepting name of Actor as input.
16. Write a short note on DOM.
17. Write a PHP script to read student.xml file which contains student roll no, name, address, college, course. Print students details of specific course in tabular format after accepting course as input.

18. Explain rules to write XML elements and attributes.

19. Write a PHP script to read book.XML and print book details in tabular format using simple XML. (Content of book.XML are (bookcode, bookname, author, publisher, price).

20. Explain any five advantages of XML over HTML.

21. Write a php script to read item-XML file (contain INo, Iname, I-desc,Price) and print item details in tabular format(using simple XML).

Answers

A.

1.	a	2.	b	3.	c
4.	d	5.	e	6.	d
7.	a	8.	a		

Questions Asked in Previous Year Exams

1 Mark

A. Fill in the blanks

1. Ajax stands for ..... [Oct. 2018]
   i. asynchronous java and XML
   ii. asynchronous javascript and XML
   iii. asynchronous jsp and XML
   iv. synchronous java and XML

2. XML–RPC stands for ..... [Oct. 2018]
   i. XML-Reverse Protocol Call
   ii. XML-Return Protocol Call
   iii. XML-Based Remote Procedure Call
   iv. None of the above

3 Which one is correct? [Oct. 2018]
   i. <MOVIE> <name> XYZ </name> </MOVIE>
   ii. <movie> <name> XYZ </movie> </name>
   iii. <MOVIE> <NAME> XYZ </name> </movie>
   iv. None of the above

B. Answer the following

1. List the example of AJAX. [Apr. 2019]

5 Marks

1. Write a note on an AJAX PHP framework. [Apr. 2019]
2. Write PHP script using AJAX concept to check username and password are valid or invalid. [Apr. 2019]
3. Explain the working of Ajax in detail. [Oct. 2018]
4. What is XML? Give any three advantages and disadvantages of XML. [Oct. 2018]
5. What is difference between XML and HTML? [Oct. 2018]
6. Explain rules to write XML elements and attributes. [Oct. 2018]

Bibliography

Reference Books

1. Programming PHP By Rasmus Lerdorf and Kevin Tatroe, O'Reilly publication
2. Beginning PHP5, Wrox publication
3. PHP for Beginners, SPD publication
4. PHP: A Beginner's Guide - Vikram Vaswani– First Edition McGraw-Hill Education
5. Learning PHP, MySQL, JavaScript, & CSS: A Step-by-Step Guide to Creating Dynamic Websites: Robin Nixon– Second Edition – O'Reilly

E-Books

1. The Complete Reference – Steven Holzner

   https://books.google.co.in/books?id=bGS4CmJY0I8C&printsec=frontcover&dq=PHP+ebook&hl=en&sa=X&ved=0ahUKEwjl4PuNoKLpAhURwTgGHXadDbYQ6AEIVTAF#v=onepage&q&f=false

2. Programming PHP – Rasmus Lerdorf , Kevin Tatroe and Peter Macintyre

   https://books.google.co.in/books?id=h-E1IVkoskC&printsec=frontcover&dq=PHP+ebook&hl=en&sa=X&ved=0ahUKEwjl4PuNoKLpAhURwTgGHXadDbYQ6AEIcDAI#v=onepage&q=PHP%20ebook&f=false

3. Beginner to Intermediate PHP5 – Mario Lurig

   https://books.google.co.in/books?id=noi76uKOJ5wC&printsec=frontcover&dq=PHP+ebook&hl=en&sa=X&ved=0ahUKEwjl4PuNoKLpAhURwTgGHXadDbYQ6AEIMDAB#v=onepage&q&f=false

4. PHP MySQL, JavaScript & HTML5 – A iley Brand

   https://books.google.co.in/books?id=p9BuBgAAQBAJ&printsec=frontcover&dq=PHP+ebook&hl=en&sa=X&ved=0ahUKEwjl4PuNoKLpAhURwTgGHXadDbYQ6AEIQTAD#v=onepage&q&f=false

Reference websites

1. http://www.php.net/
2. https://www.w3schools.com/php/
3. https://www.tutorialspoint.com/php/
4. https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html
5. https://zetcode.com/lang/php/lexis/