

UNIT-V
CLIENT SIDE
SCRIPTING

*

Introduction to JavaScript :

JavaScript is a dynamic language that executes within a browser. JavaScript code is embedded within an HTML page using the JavaScript tag. The `<script>` tag is used to embed JavaScript code. JavaScript code can be embedded in :

- An external file.
- The header of the page
- The body of the page.

* In this example, JavaScript is embedded within the header. As soon as the page is loaded this code is executed.

```
<html>
```

```
<head>
```

```
<title>JavaScript Example </title>
```

```
<script language = " JavaScript 1.2">
```

```
<!--
```

```
document.write ("Hello world");
```

```
//-->
```

```
</script>
```

```
</head>
```

```
</html>
```

```
<body> The body </body>
```

```
</html>
```

* The Document write method displays the text.

Hello world The body

* Notice that the JavaScript code is enclosed in HTML commented tags.

<!--

//-->

These are often used to surround JavaScript code. In older browsers JavaScript was not recognized or handled. To avoid the display of this code in a page, the browser would ignore the contents of the comment. However, in a browser that supports JavaScript the comments tags are ignored and the code is executed.

Uses of JavaScript :

* JavaScript can be used as an alternative to Java applets.

* JavaScript can get embedded in XHTML.

* Using DOM JavaScript can access and modify the properties of CSS (cascading style sheets) and contents of XHTML document.

* JavaScript can be used to detect the visitor's browsers and can load the page accordingly

(2)

* JavaScript can be used to create cookies.

Features of JavaScript :

- 1) Browser support : For running the JavaScript in the browser there is no need to use some plug-in. Almost all the popular browsers support JavaScripting.
- 2) It automatically inserts the semicolon at the end of the statement, hence there is no need to write semicolon at the end of the statement in JavaScript.
- 3) Dynamic Typing : It supports, dynamic typing, that means the data type is bound to the value and not to the variable.
- 4) Run Time Evaluation : Using the 'eval' function the expression can be evaluated at runtime.
- 5) Support for Object : JavaScript is object oriented scripting language. JavaScript has a small number of in-built objects.
- 6) Function Programming : In JavaScript functions are used. One function can accept another function as a parameter.

* JavaScript Variable

Variable means anything that can vary. JavaScript includes variables which hold the data value and it can be changed anytime.

* JavaScript uses reserved keyword "var" to declare a variable. A variable must have a unique name. You can assign a value to a variable using equal to (=) operator when you declare it or before using it.

Syntax:

```
var <variable-name>;
```

```
var <variable-name> = <value>;
```

Example : Variable declaration & Initialization

```
var one = 1; // variable stores numeric value
```

```
var two = 'two'; // variable stores string value
```

```
var three; // declared a variable without assigning a value
```

In the above example, we have declared three variables using var keyword: one, two and three. we have assigned values to variables one and two at the same time when we declared it, whereas variable three is declared but does not hold any value yet, so its value will be 'undefined'.

Declare variables in Single line:

Multiple variables can also be declared in a single line separated by comma.

Example: Multiple Variables in Single Line

```
var one = 1, two = 'two', three;
```

Declare variable without var keyword:

JavaScript allows variable declaration without var keyword. You must assign a value when you declare a variable without var keyword.

Example: Variable without var keyword

```
one = 1;
```

* Scope of the variables declared without var keyword become global irrespective of where it is declared. Global variables can be accessed from anywhere in the webpage.

Loosely-typed Variables:

C# or Java has strongly typed variables. It means variable must be declared with a particular data type, which tells what type of data the variable will hold.

JavaScript variables are loosely-typed which means it does not require a data type to be declared. You can assign any type of literal values to a variable i.e., string, integer, float, boolean etc..

Example : Loosely Typed Variables

var one = 1; // numeric value

one = 'one'; // string value

one = 1.1; // decimal value

one = true; // Boolean value

one = null; // null value

Primitive Types :

JavaScript defines two entities primitives and objects. The primitives are for storing the values whereas the object is for storing the reference to the actual value.

* There are following primitive types used in JavaScript.

- 1) Number
- 2) String
- 3) Boolean
- 4) Undefined
- 5) Null

* There are three types of predefined objects in JavaScript.

- 1) Number
- 2) String
- 3) Boolean

These objects are called wrapper objects. These wrapper objects provide properties and methods which can be used by primitive types.

* Scope Of Variables:

Scope in JavaScript defines accessibility of variables, objects and functions.

* There are two types of scope in JavaScript

- 1) Global scope
- 2) Local scope.

1) Global Scope:

Variables declared outside of any function become global variables. Global variables can be accessed and modified from any function.

Example: Global scope

```
<script>
```

```
var userName = "Peter";
```

```
function modifyUserName ( )
```

```
{
```

```
    userName = "Steve";
```

```
};
```

```
function showUserName ( )
```

```
{
```

```
    alert (userName);
```

```
};
```

```
alert (userName); // display peter
```

```
modifyUserName ( );
```

```
showUserName ( ); //display steve
```

```
</script>
```


(5)

* In the above example, the variable `userName` becomes a global variable because it is declared outside of any function. A `modifyUserName()` function modifies `userName` as `userName` is a global variable and can be accessed inside any function. The same way, `showUserName()` function displays current value of `userName` variable. Changing value of global variable in any function will reflect throughout the program.

* Note that variables declared inside a function without `var` keyword also become global variables.

2) Local Scope:

Variables declared inside any function with `var` keyword are called local variables. Local variables cannot be accessed or modified outside the function declaration.

Example: Local scope

```
<script>
```

```
function createUserName()  
{  
    var userName = "Peter";
```

```
}  
function showUserName()  
{  
    alert(userName);  
}
```

```
createUserName();  
showUserName(); // throws error: userName  
</script> is not defined.
```

* In the above example, `userName` is local to `createUserName()` function. It cannot be accessed in `showUserName()` function or any other functions. It will throw an error if you try to access a variable which is not in the local or global scope. Use try catch block for exception handling.

No Block level Scope :

JavaScript does not allow block level scope inside `{ }`. For example, variables defined in `if` block can be accessed outside `if` block, inside a function.

Example : No block level scope

```
Function NoBlockLevelScope()  
{  
  if (1 > 0)  
  {  
    var myVar = 22;  
  }  
  alert(myVar);  
}
```

`NoBlockLevelScope();`

(6)

* Following are the variable scoping rules used in JavaScript.

1) Script level scope :

If a variable is declared with a `var` and if it is declared outside any function then it has the script level scope. This variable is also called "global variable".

2) Function level scope :

If a variable is declared with a `var` inside a function then it has at the function level scope. A variable with a function level scope, called "local variable".

3) Auto-declaration :

If a variable is used without the `var` declaration statement, it will be automatically declared with the script level scope, becoming a global variable. But using this approach of auto declaration global variables is not recommended.

4) Collision :

If a variable is explicitly defined in a function has the same name as the variable defined outside the function, then the variable outside the function cannot be accessible within this function.

*

Functions

7

A function consists of the function keyword followed by the name of the function, a set of open and close parentheses enclosing an optional parameter list and a body enclosed in a set of curly braces.

Syntax:

```
function functionName(parameterList)
{
    //body
}
```

* A function uses a return keyword to return a value from a function.

```
<html>
<head>
    <title> JavaScript Example </title>
    <script type = "text/javascript">
        function getHeader()
        {
            return "<h1> Main Heading </h1>"
        }
    </script>
</head>
<body>
    <script type = "text/javascript">
        document.write (getHeader());
    </script>
</body>
</html>
```


o/p:

JavaScript Example
Main Heading

* Parameters are separated by commas in the function declaration.

```
<html>
```

```
<head>
```

```
<title> JavaScript Example </title>
```

```
<script type = "text/javascript">
```

```
function multiply (num1, num2)
```

```
{
```

```
    return num1 * num2;
```

```
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<script type = "text/javascript">
```

```
document.write (multiply (2, 4));
```

```
</script>
```

```
</body>
```

```
</html>
```

o/p:

JavaScript Example.
8

Passing an array to the function:

8

Similar to C or C++ we can pass an entire array as a parameter to the function. This method of array passing is called "call by reference".

Example:

```
<html>
```

```
<head>
```

```
<script type="text/javascript">
```

```
function display(a)
```

```
{
```

```
    document.write("The contents of the array  
are .. "+ "<br>");
```

```
    i=0;
```

```
    for(i in a)
```

```
    {
```

```
        document.write(a[i]+"<br>");
```

```
        i++;
```

```
    }
```

```
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<script type="text/javascript">
```

```
var ar = new Array(10);
```

```
for(i=0; i<=9; i++)
```

```
{
```

```
    ar[i]=i;
```

```
}
```

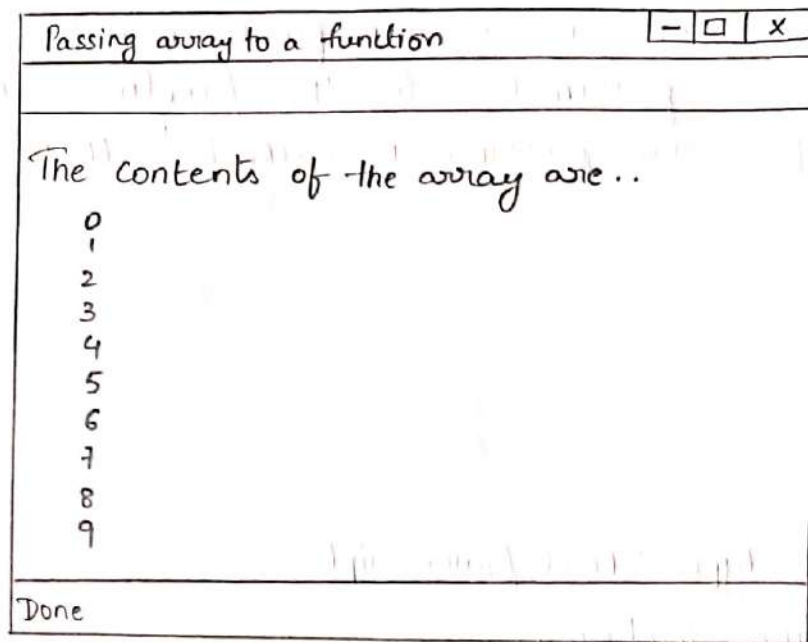
```
display(ar);
```

```
</script>
```

```
</body>
```

```
</html>
```

Output :



*

Event's Handlers

9

Event is an activity that represents a change in the environment. For example, mouse clicks, pressing a particular key on keyboard represent the events. Such events are called "intrinsic events".

* Event handler is a script that gets executed in response to these events. Thus event handler enables the web document to respond the user activities through the browser window.

* Events are specified in lowercase letters and these are case-sensitive.

* The process of connecting event handler to an event is called event registration. The event handler registration can be done using two methods -

- Assigning the tag attributes.
- Assigning the handler address to object properties.

Events, Attributes and Tags:

On occurrence of events the tag attribute must be assigned with some used defined functionalities. This will help to execute certain action on occurrence of particular event.

Commonly used events and tag attributes are enlisted in the following table -

Events	Intrinsic event attribute	Meaning	Associated tags
change	onchange	on occurrence of some change	<input> <textarea> <select>
click	onclick	when user clicks the mouse button	<a> <input>
mouseout	onmouseout	when the user moves the mouse away from some element	Form elements such as input, button, text, textarea & soon
mouseover	onmouseover	when the user moves the mouse away over some element	Form elements such as input, button, text, textarea and so on.
load	onload	After getting the document loaded	<body>
reset	onreset	when the reset button is clicked	<form>
submit	onsubmit	when the submit button is clicked	<form>
select	onselect	on selection	<input> <textarea>

Handling Events from the Body Elements

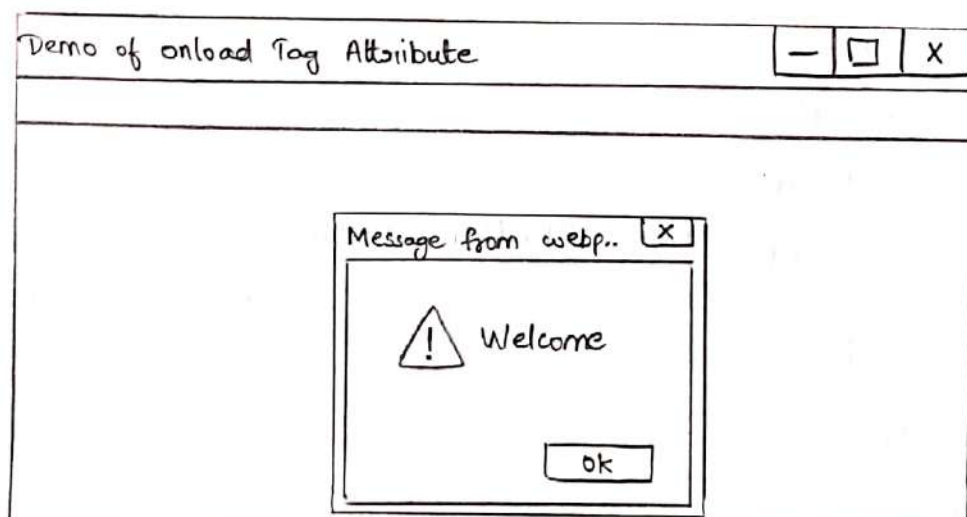
10

To understand how events works in JavaScript let us put some form components on the JavaScript. The onload event gets activated as soon as the web page gets loaded in the browser's window. Following script along with the output illustrate the onload tag attribute

Onload Demo.html

```
<html>
  <head>
    <script type="text/javascript">
      function my-fun()
      {
        alert("Welcome");
      }
    </script>
  </head>
  <body onload="my-fun">
  </body>
</html>
```

Output



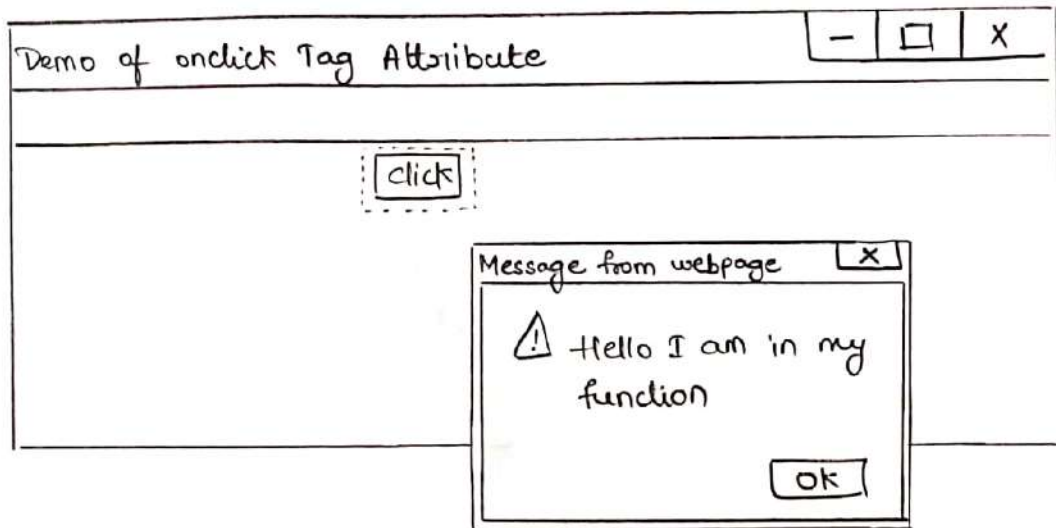
Handling Events from Button Elements :

For handling the event using button element we have used the tag attribute onclick. The idea is that whenever we click the button some event handler must be called. This event handler can be a user defined function in which certain set of instructions get executed.

* Following is a simple JavaScript in which on the button click we have called a function `my-func()`. This is a simple function in which we have displayed some message using alert popup box.

OnClickDemo.html

```
<html>
<head>
  <title>Demo of onclick Tag Attribute </title>
  <script type="text/javascript">
    function my-func()
    {
      alert("Hello I am in my function");
    }
  </script>
</head>
<body>
  <form>
    <input type="button" value="click"
      onclick="my-func()" />
  </form>
</body>
</html>
```



* Document Object Model:

The Document Object Modeling (DOM) is for defining the standard for accessing and manipulating XHTML, XML and other scripting languages.

* Basically, DOM is an Application Programming Interface (API) that defines the interface between XHTML document and application program. That means, suppose application program is written in java and this java program wants to access the elements of XHTML web document then it is possible by using a set of API which belongs to the DOM.

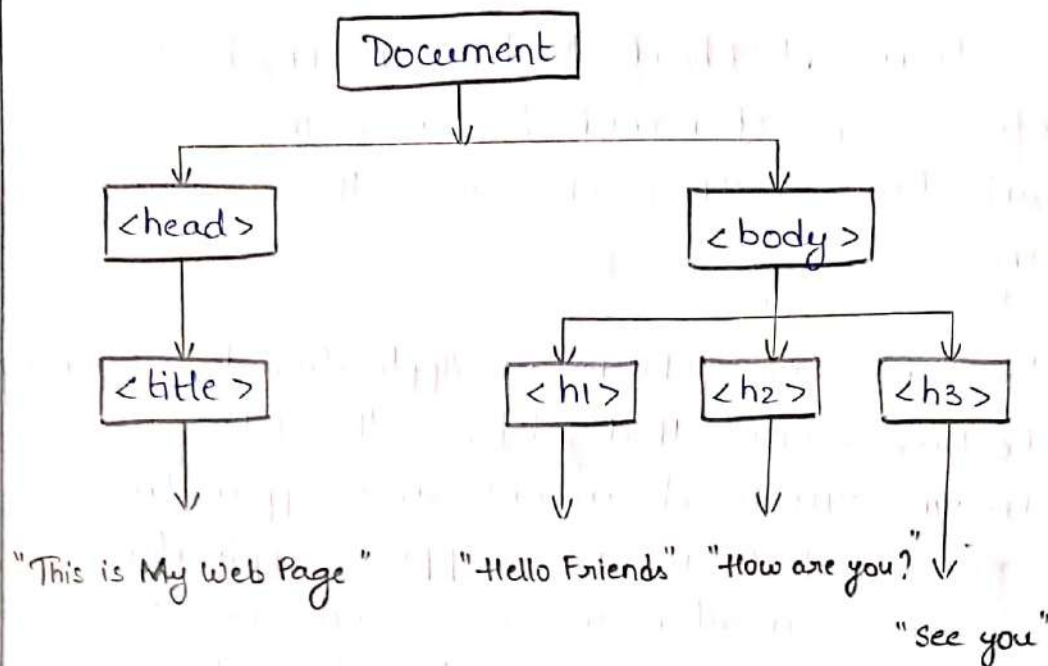
DOM Tree:

The documents in DOM are represented using a tree like structure in which every element is represented as a node. Hence the tree structure is also referred as DOM tree.

Example:

```
<html>
  <head>
    <title>This is My Web Page </title>
  </head>
  <body>
    <h1>Hello Friends </h1>
    <h2>How are you ? </h2>
    <h3>See you </h3>
  </body>
</html>
```

The DOM tree will be



* We can describe some basic terminologies used in DOM tree as follows—

- 1) Every element in the DOM tree is called node.
- 2) The topmost single node in the DOM tree is called the root.
- 3) Every child node must have a parent node.
- 4) The bottommost nodes that have no children are called leaf nodes.
- 5) The nodes that have the common parent are called siblings.

DOM Tree Traversal and Modification:

The main intension of objects is to use the collection of all related objects on the web page. There is a special object model collection called 'all' which is used to refer all the HTML documents. The order in which these HTML elements come in our program in the same order all

those elements will be displayed.

Example:

```
<html>
```

```
<head>
```

```
<script type="text/javascript">
```

```
var web-page-element = " ";
```

```
function Display()
```

```
{
```

```
for (i=0; i<document.all.length; i++)
```

```
web-page-element += "<br>"+  
document.all[i].tagName;
```

```
pmg.innerHTML += web-page-element;
```

```
}
```

```
</script>
```

```
</head>
```

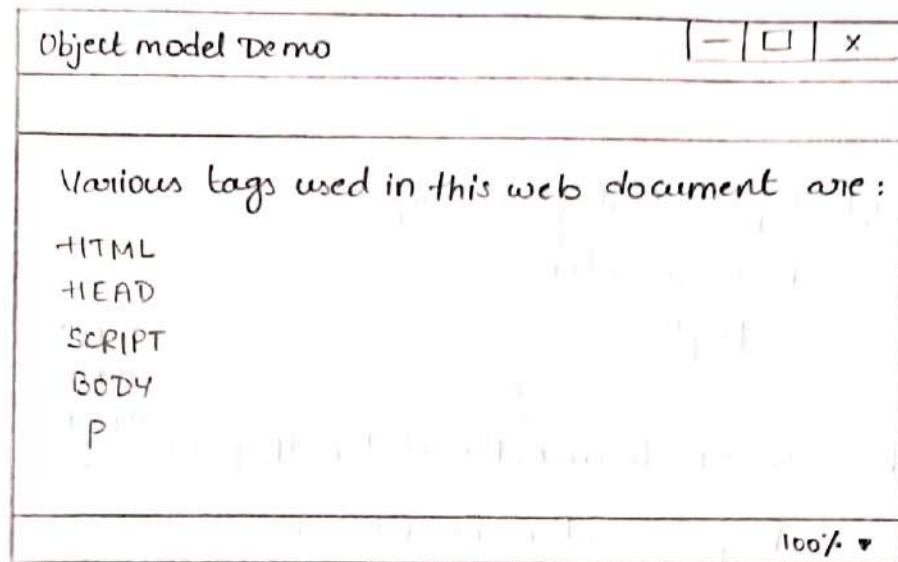
```
<body onload="Display()">
```

```
<p id="pmg"> Various tags used in this web  
document are: </p>
```

```
</body>
```

```
</html>
```

Output :



*

Form Validation

(14)

Form Validation is a technique which is useful in checking the validity of the input submitted by the user. One of the common technique used in form validation is password validation.

* We can use password verification process that comes along the web document. In this process user has to enter the password correctly for two times. If both the passwords are matching then the password is verified. Normally this facility is given when user creates his web account.

* In the following Javascript we have used two textboxes which are of password type. That means whatever we type in these boxes appear in the form of dots. we will compare the entries in the two text boxes; if those are not same then the alert message will be displayed.

Example: TextDemo.html

```
<html>
```

```
<head>
```

```
<script type = "text/javascript">
```

```
function my-fun()
```

```
{
```

```
var mypwd = document.getElementById  
("pwd");
```

```
var my-re-pwd = document.getElementById
```

```
By Id ("re-pwd");
```

```
if (mypwd.value == " ")
```

```
{
```

```
    alert ("You have not entered the  
password");
```

```
    mypwd.focus();
```

```
    return false;
```

```
}
```

```
if (mypwd.value != my-re-pwd.value)
```

```
{
```

```
    alert ("Password is not verified ,  
Re-enter both the passwords");
```

```
    mypwd.focus();
```

```
    mypwd.select();
```

```
    return false;
```

```
}
```

```
else
```

```
{
```

```
    alert ("Congratulations !!!");
```

```
    return true;
```

```
}
```

```
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<form id = "form1">
```

```
<label> Enter your password
```

```
<input type = "password" value = " "  
id = "pwd" />
```

```
</label>
```

```
<br/> <br/>
```

<label> Re-Enter the password (15)

<input type="password" value=""
id="re_pwd" onblur="my-fun();" />

</label>

<input type="submit" value="submit"
name="submit" onsubmit=
"my-fun();" />

<input type="reset" value="Reset"
name="reset" />

</form>

~~</html>~~ </body>
</html>

Simple AJAX Application

AJAX is a Asynchronous Java Script. It is not a new programming language but it is a kind of web document which adopts certain standards. AJAX allows the developer to exchange the data with the server and updates the part of web document without reloading the webpage.

How AJAX Works:

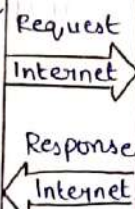
When user makes a request, the browser creates a object for the XMLHttpRequest and a request is made to the server over an internet. The server processes this request and sends the required data to the browser. At the browser side the returned data is processed using JavaScript and the web document gets updated accordingly.

web browser

- 1) User makes a request
- 2) XMLHttpRequest object is created
- 3) Sends Http request
- 6) Processes the returned data.
- 7) Update web document

server

- 4) Processes the Http request
- 5) Create a response and send data to browser.



Let us understand how AJAX works with the help of an example.

```
<html>
```

```
<head>
```

```
<script type="text/javascript">
```

```
function Myfun()
```

```
{
```

```
    if (window.XMLHttpRequest)
```

```
    {
```

```
        req = new XMLHttpRequest();
```

```
    }
```

```
    else
```

```
    {
```

```
        req = new ActiveXObject("Microsoft.  
                                req");
```

```
    }
```

```
    req.onreadystatechange = function()
```

```
    {
```

```
        if (req.readyState == 4 &&
```

```
            req.status == 200)
```

```
        {
```

```
            document.getElementById
```

```
                ("myID").innerHTML =
```

```
                req.responseText;
```

```
        }
```

```
    }
```

```
    req.open("GET", "newdata.txt", true);
```

```
    req.send();
```

```
}
```

```
</script>
```

```
</head>
```

<body>

(17)

<div id = "my ID"> This text can be changed
</div>

< button type = " button" onclick = " MyFun()">
Change </button>

</body>

</html>

In above script, we have written some text which can be replaced by some another text on button click. On button click a function Myfun is invoked. In this function

1. XMLHttpRequest object is used to exchange data with a server. This object allows the user to change / update the parts of the web page without reloading it fully. The modern web browsers such as IE-7+, Firefox, Chrome have built in XMLHttpRequest but old web browsers make use of ActiveXObject.

2. when a request to a server is sent, then onreadystatechange event is triggered.

• The readyState property holds the status of the XMLHttpRequest. The readyState = 4 means request is finished and response is ready. The status = 200 means "ok".

3. The request can be sent to the server by using two functions `open()` and `send()`.

```
req.open("GET", "newdata.txt", true);
```

GET or POST
method

Location of
file on server

true: means
asynchronous

false: means
synchronous

- * Asynchronous communication allows fast processing of the data.

- * The `newdata.txt` file contains some updating text using which our web page can be updated.

- * The `send()` method sends the request to the server.

AJAX with XML and PHP:

We can use Ajax along with XML and PHP. In the following example we will discuss how, HTML file along with javascript communicates with XML and PHP. It will work as follows -

1. HTML displays the form that contains a drop down list. User can select his/her friend's name from the dropdown list.

2. when user selects some name, a function named `showNames` will be triggered. This function is defined in javascript file.

3. This function in javascript file will send ⁽¹⁸⁾ the name as a query string to some php file. The name of the PHP file as considered as wil.

4. The PHP file make use of DOM. It will load XML file using DOM. Using DOM object we go through each node of XML file and

5. These contents are then returned to the HTML using the innerHTML. Hence on browser we can get the details of the friend whose name we have selected.

Step:1

Create an HTML document for displaying the form.

AjaxDemo.html

<html>

<head>

<script src = "testing.js" > </script>

</head>

<body>

<form>

Select a name :

<select name = "names" onchange = "show
Names(this.value)">

<option value = "chitra" > Chitra </option>

<option value = "Priyanka" > Priyanka
</option>


```

        <option value = "Raj"> Raj </option>
    </select>
</form>
<p>
    <div id = "textHint"><b> Friend Details :
                                </b></div>
    </p>
</body>
</html>

```

Step: 2

The javascript will be as follows. It contains the function showNames.

testing.js

```

var xmlhttp;
function showNames(str)
{
    xmlhttp = GetXmlHttpObject();
    {
        alert("Browser does not support HTTP
                Request");
        return;
    }
    var url = "getInfo.php";
    url = url + "?q=" + str;
    url = url + "&sid=" + Math.random();
    xmlhttp.onreadystatechange = stateChanged;
    xmlhttp.open("GET", url, true);
    xmlhttp.send(null);
}

```

```
function stateChanged()
```

```
{
    if (xmlHttp.readyState == 4 || xmlHttp.status
        == 200)
```

```
{
    document.getElementById("txtHint").
        innerHTML = xmlHttp.responseText;

```

```
}
```

```
}
```

```
function GetXmlHttpRequest()
```

```
{
```

```
    var xmlHttp = null;
```

```
    try
```

```
    {
```

```
        // Firefox, Opera 8.0+, Safari
```

```
        xmlHttp = new XMLHttpRequest();
```

```
    }
```

```
    catch (e)
```

```
    {
```

```
        // Internet Explorer
```

```
        try
```

```
        {
```

```
            xmlHttp = new ActiveXObject
                ("Msxml2.XMLHTTP");
```

```
        }
```

```
        catch (e)
```

```
        {
```

```
            xmlHttp = new ActiveXObject
                ("Microsoft.XMLHTTP");
```

```
        }
```

```
    }
```

```
    return xmlHttp;
```

```
}
```

Step: 3

The PHP script that normally runs on the server side is as given below. It will make use of DOM to load and handle the XML file.

getInfo.php

<?php

```
$q = $_GET["q"]
```

```
$xmlDoc = new DOMDocument();
```

```
$xmlDoc → load("FriendNames.xml");
```

```
$a = $xmlDoc → getElementByTagName('name');
```

```
for ($i=0; $i <= $a → length-1; $i++)
```

```
{  
    if ($a → item($i) → nodeType == 1)
```

```
    {  
        if ($a → item($i) → childNodes → item(0)  
            → nodeValue == $q)
```

```
        {  
            $b = ($a → item($i) → parentNode);
```

```
        }
```

```
    }
```

```
}
```

```
$f = ($b → childNodes);
```

```
for ($i=0; $i < $f → length; $i++)
```

```
{
```

```
    if ($f → item($i) → nodeType == 1)
```

```
    {  
        echo("<b>". $f → item($i) → nodeName.  
            : "</b>");
```

```
        echo($f → item($i) → childNodes → item(0)  
            → nodeValue);
```

```
        echo("<br />");
```

```
    }
```

```
}
```

Step: 4

(20)

The XML file which is handled by the PHP in above step is

FriendNames.xml

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

```
<Friend>
```

```
  <Info>
```

```
    <name> Chitra </name>
```

```
    <phone> 1111111111 </phone>
```

```
    <email> Chitra-abc@gmail.com </email>
```

```
    <hobby> Singing </hobby>
```

```
  </Info>
```

```
  <Info>
```

```
    <name> Priyanka </name>
```

```
    <phone> 2222222222 </phone>
```

```
    <email> Pri123@rediffmail.com </email>
```

```
    <hobby> Reading </hobby>
```

```
  </Info>
```

```
  <Info>
```

```
    <name> Raj </name>
```

```
    <phone> 3333333333 </phone>
```

```
    <email> Raj-2008@hotmail.com </email>
```

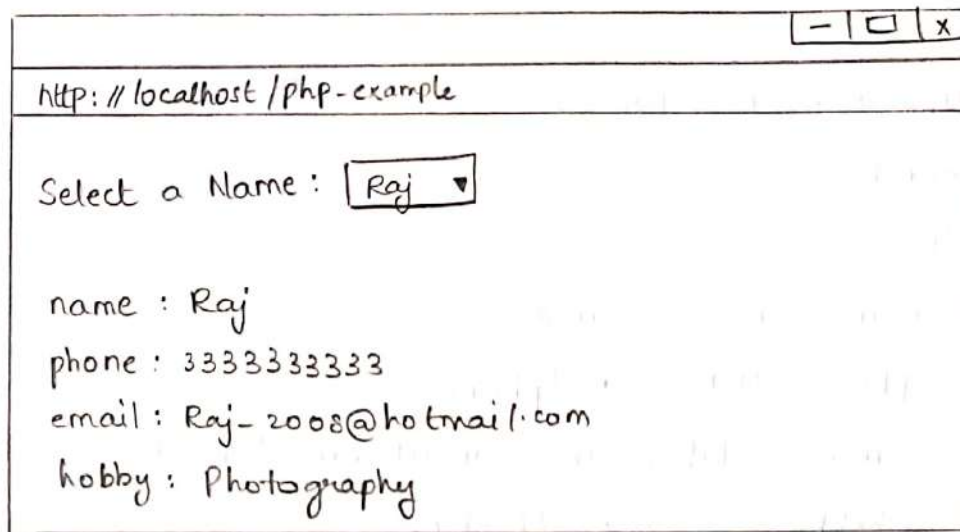
```
    <hobby> Photography </hobby>
```

```
  </Info>
```

```
</Friend>
```


Step: 5

For getting the output we will open the HTML file (created in step 1) in browser window



A hand-drawn representation of a web browser window. The title bar at the top contains three icons: a minus sign, a square, and an 'x'. Below the title bar is the address bar, which contains the text 'http://localhost/php-example'. The main content area of the browser displays the output of a web form. It starts with the text 'Select a Name:' followed by a dropdown menu that has 'Raj' selected and a small downward arrow. Below this, there are four lines of text: 'name : Raj', 'phone : 3333333333', 'email : Raj-2008@hotmail.com', and 'hobby : Photography'.

http://localhost/php-example

Select a Name:

name : Raj
phone : 3333333333
email : Raj-2008@hotmail.com
hobby : Photography