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

NEW SYLLABUS
CBCS PATTERN

WEB TECHNOLOGIES-II

Dr. A. B. NIMBALKAR

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

SPPU New Syllabus

A Book Of

WEB TECHNOLOGIES – II

For T.Y.B.Sc. Computer Science : Semester – VI

[Course Code CS 363 : Credits – 2]

CBCS Pattern

As Per New Syllabus, Effective from June 2021

Dr. A. B. Nimbalkar

M.C.S., M.Phil. D.C.L, Ph.D. (Comp. Sci.)

Asst. Professor, Department of Computer Science

Annasaheb Magar Mahavidyalaya, Hadapsar

Pune

Price ₹ 270.00

NIRALI PRAKASHAN

ADVANCEMENT OF KNOWLEDGE

N5948

WEB TECHNOLOGIES - II **ISBN 978-93-5451-325-1**

First Edition : January 2022

Published By :
NIRALI PRAKASHAN
Abhyudaya Pragati, 1312, Shivaji Nagar,
Off J.M. Road, Pune – 411005
Tel - (020) 25512336/37/39
Email : niralipune@pragationline.com

Polyplate

Printed By :
YOGIRAJ PRINTERS AND BINDERS
Survey No. 10/1A, Ghule Industrial Estate
Nanded Gaon Road
Nanded, Pune - 411041

DISTRIBUTION CENTRES

PUNE

Nirali Prakashan
(For orders outside Pune)
S. No. 28/27, Dhayari Narhe Road, Near Asian College
Pune 411041, Maharashtra
Tel : (020) 24690204; Mobile : 9657703143
Email : bookorder@pragationline.com

Nirali Prakashan
(For orders within Pune)
119, Budhwar Peth, Jogeshwari Mandir Lane
Pune 411002, Maharashtra
Tel : (020) 2445 2044; Mobile : 9657703145
Email : niralilocal@pragationline.com

MUMBAI

Nirali Prakashan
Rasdhara Co-op. Hsg. Society Ltd., 'D' Wing Ground Floor, 385 S.V.P. Road
Girgaum, Mumbai 400004, Maharashtra
Mobile : 7045821020, Tel : (022) 2385 6339 / 2386 9976
Email : niralimumbai@pragationline.com

DISTRIBUTION BRANCHES

DELHI

Nirali Prakashan
Room No. 2 Ground Floor
4575/15 Omkar Tower, Agarwal Road
Darya Ganj, New Delhi 110002
Mobile : 9555778814/9818561840
Email : delhi@niralibooks.com

BENGALURU

Nirali Prakashan
Maitri Ground Floor, Jaya Apartments,
No. 99, 6th Cross, 6th Main,
Malleswaram, Bengaluru 560003
Karnataka; Mob : 9686821074
Email : bengaluru@niralibooks.com

NAGPUR

Nirali Prakashan
Above Maratha Mandir, Shop No. 3,
First Floor, Rani Jhanshi Square,
Sitabuldi Nagpur 440012 (MAH)
Tel : (0712) 254 7129
Email : nagpur@niralibooks.com

KOLHAPUR

Nirali Prakashan
438/2, Bhosale Plaza, Ground Floor
Khasbag, Opp. Balgopal Talim
Kolhapur 416 012, Maharashtra
Mob : 9850046155
Email : kolhapur@niralibooks.com

JALGAON

Nirali Prakashan
34, V. V. Golani Market, Navi Peth,
Jalgaon 425001, Maharashtra
Tel : (0257) 222 0395
Mob : 94234 91860
Email : jalgaon@niralibooks.com

SOLAPUR

Nirali Prakashan
R-158/2, Avanti Nagar, Near Golden
Gate, Pune Naka Chowk
Solapur 413001, Maharashtra
Mobile 9890918687
Email : solapur@niralibooks.com

marketing@pragationline.com | www.pragationline.com

Also find us on www.facebook.com/niralibooks

Preface ...

I take an opportunity to present this Text Book on **"Web Technologies - II"** to the students of Third Year B.Sc. (Computer Science) Semester-VI as per the New Syllabus, June 2021.

The book has its own unique features. It brings out the subject in a very simple and lucid manner for easy and comprehensive understanding of the basic concepts. The book covers theory of Introduction to Web Techniques, XML, JavaScript and jQuery, Ajax, and PHP Framework CodeIgniter.

A special word of thank to Shri. Dineshbhai Furia, and Mr. Jignesh Furia for showing full faith in me to write this text book. I also thank to Mr. Amar Salunkhe and Mr. Akbar Shaikh of M/s Nirali Prakashan for their excellent co-operation.

I also thank Ms. Chaitali Takle, Mr. Ravindra Walodare, Mr. Sachin Shinde, Mr. Ashok Bodke, Mr. Moshin Sayyed and Mr. Nitin Thorat.

Although every care has been taken to check mistakes and misprints, any errors, omission and suggestions from teachers and students for the improvement of this text book shall be most welcome.

Author

■■■

Syllabus ...

1. **Introduction to Web Techniques** **(6 Lectures)**
   - Variables
   - Server Information
   - Processing Forms
   - Setting Response Headers
   - Maintaining State
   - PHP Error Handling
2. **XML** **(6 Lectures)**
   - What is XML?
   - XML Document Structure
   - PHP and XML
   - XML Parser
   - The Document Object Model
   - The Simple XML Extension
   - Changing a Value with SimpleXML
3. **JavaScript and jQuery** **(10 Lectures)**
   - Overview of JavaScript
   - Object Orientation and JavaScript Basic Syntax (JS Datatypes, JS Variables)
   - Primitives, Operations and Expressions
   - Screen Output and Keyboard Input (Verification and Validation)
   - JS Control Statements and JS Functions
   - JavaScript HTML DOM Events (onmouseup, onmousedown, onclick, onload, onmouseover, onmouseout)
   - JS Popup Boxes (Alert, Confirm, Prompt)
   - Jquery Library, Including Jquery Library in Page
   - Jquery Selector, DOM Manipulation using jQuery
4. **AJAX** **(6 Lectures)**
   - Introduction of AJAX
   - AJAX Web Application Model
   - AJAX - PHP Framework
   - Performing AJAX Validation
   - Handling XML Data using PHP and AJAX
   - Connecting Database using PHP and AJAX
5. **PHP Framework CodeIgniter** **(8 Lectures)**
   - CodeIgniter - Overview, Installing CodeIgniter
   - Application Architecture
   - MVC Framework, Basic Concept of CodeIgniter, Libraries
   - Working with Databases
   - Load External JS and CSS Page and Redirecting from Controller, Adding JS and CSS, Page Redirection
   - Loading Dynamic Data on Page and Session Management, Cookies Management

■■■

Contents ...

■■■

CHAPTER 1

Introduction to Web Techniques

Objectives...

- To understand Server Variables Concepts
- To study Server Information
- To learn Forms and Processing of Form
- To Setting Response Headers and Maintaining State
- To understand Error Handling in PHP

1.0 INTRODUCTION

- Nowadays, PHP is becoming a standard in the world of Web programming with its simplicity, performance, reliability, flexibility and speed.
- PHP was designed/developed as a Web scripting language and although it is possible to use it in purely command-line and GUI scripts, the Web accounts for the vast majority of PHP uses.
- A dynamic Web site may have forms, sessions, and sometimes redirection. This chapter explains how to implement those things in PHP.
- The Web runs on the HyperText Transfer Protocol (HTTP) and this protocol governs how web browsers request files from web servers and how the servers send the files back.
- Web technology is essential today because Internet has become the number one source to information and many of the traditional software applications have become Web Applications.

1.1 VARIABLES [Oct. 17, April 18]

- Every time we click a link or submit a form, we send a great deal of data about the system and the browser to the Web server and each time the web server responds, it sends us a great deal of data about itself. PHP has the capability to capture that data.
- For instance, if we go to a PHP driven website and log in, it's likely that a predefined variable $_POST is filled (on server) with the username and password, another predefined variable named $_SERVER contains information about the current Web server environment.

- PHP automatically have some variables called predefined variables available anywhere in the program.
- They are array variables. These variables are: $_ENV, $_GET, $_POST, $_COOKIE, and $_SERVER, referred to as EGPCS.
- There is a setting in the configuration file (php.ini) called register_globals. The default value is off and it restricts how we can access some predefined variables.
- The register_globals determine whether or not to register the EGPCS variables as global variables.
- Regardless of the setting of the option register_globals, PHP creates six global arrays that contain the EGPCS information as given below:
  1. **$_COOKIE**: This global array contains any cookie values passed as part of the request, where the keys of the array are the names of the cookies.
  2. **$_GET**: This global array contains any parameters that are part of a GET request, where the keys of the array are the names of the form parameters.
  3. **$_POST**: This global array contains any parameters that are part of a POST request, where the keys of the array are the names of the form parameters.
  4. **$_FILES**: This global array contains information about any uploaded files. **[Oct. 17, April 18]**
  5. **$_SERVER**: This global array contains useful information about the web server, as described in the next section. **[April 19]**
  6. **$_ENV**: This global array contains the values of any environment variables, where the keys of the array are the names of the environment variables.
- Above variables are not only global, but also visible from within function definitions, unlike their longer counterparts.
- The $_REQUEST array is also created by PHP if the register_globals option is on. The $_REQUEST array contains the elements of the $_GET, $_POST, and $_COOKIE arrays.
- PHP also creates a variable called $PHP_SELF, which holds the name of the current script, relative to the document root.
- Several predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope and we can access them from any function, class or file without having to do anything special.
- The PHP superglobal variables are $GLOBALS, $_SERVER, $_REQUEST, $_POST, $_GET, $_FILES, $_ENV, $_COOKIE, and $_SESSION.

1.2 SERVER INFORMATION [April 18]

- The $_SERVER is an associative array contains a lot of information from the Web server.
- The following table lists the most important elements that can go inside $_SERVER:

Sr. No.	Element/Code	Description
1.	$_SERVER['PHP_SELF']	Returns the filename of the currently executing script.
2.	$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common Gateway Interface (CGI) the server is using.
3.	$_SERVER['SERVER_ADDR']	Returns the IP address of the host server.
4.	$_SERVER['SERVER_NAME']	Returns the name of the host server (such as www.nirali.com).
5.	$_SERVER['SERVER_SOFTWARE']	Returns the server identification string (such as Apache/2.2.24).
6.	$_SERVER['SERVER_PROTOCOL']	Returns the name and revision of the information protocol (such as HTTP/1.1).
7.	$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST).
8.	$_SERVER['REQUEST_TIME']	Returns the timestamp of the start of the request (such as 1377687496).
9.	$_SERVER['QUERY_STRING']	Returns the query string if the page is accessed via a query string.
10.	$_SERVER['HTTP_ACCEPT']	Returns the Accept header from the current request.
11.	$_SERVER['HTTP_ACCEPT_CHARSET']	Returns the Accept_Charset header from the current request (such as utf-8,ISO-8859-1).
12.	$_SERVER['HTTP_HOST']	Returns the Host header from the current request.
13.	$_SERVER['HTTP_REFERER']	Returns the complete URL of the current page (not reliable because not all user-agents support it).
14.	$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol.

contd. ...

15.	$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page.
16.	$_SERVER['REMOTE_HOST']	Returns the Host name from where the user is viewing the current page.
17.	$_SERVER['REMOTE_PORT']	Returns the port being used on the user's machine to communicate with the web server.
18.	$_SERVER['SCRIPT_FILENAME']	Returns the absolute pathname of the currently executing script.
19.	$_SERVER['SERVER_ADMIN']	Returns the value given to the SERVER_ADMIN directive in the web server configuration file (if your script runs on a virtual host, it will be the value defined for that virtual host) (such as someone@w3schools.com).
20.	$_SERVER['SERVER_PORT']	Returns the port on the server machine being used by the web server for communication (such as 80).
21.	$_SERVER['SERVER_SIGNATURE']	Returns the server version and virtual host name which are added to server-generated pages.
22.	$_SERVER['PATH_TRANSLATED']	Returns the file system based path to the current script.
23.	$_SERVER['SCRIPT_NAME']	Returns the path of the current script.
24.	$_SERVER['SCRIPT_URI']	Returns the URI of the current page.

1.3 PROCESSING FORMS

- Forms are essential parts in web development. Forms are used to communicate between users and the server.
- Form is a way to get information of the user to the server and let the server do something in response to the user's input.
- We use form to register in a website, to login, to send feedback, to place order online, to book ticket online etc.
- After submitting the form, the form is processed by PHP on the server. In PHP the form parameters are available in the $_GET and $_POST arrays.
- When user submits a form then the form parameter is send to the server. Before sending the information, browser encodes it using a scheme called URL encoding.
- In this scheme, name/value pairs are joined with equal signs (=) and different pairs are separated by the ampersand (&).

```
name1=value1&name2=value2&name3=value3
```

- Spaces are removed and replaced with the + character and any other non alphanumeric characters are replaced with a hexadecimal values. After the information is encoded it is sent to the server.
- There are two HTTP methods that a client can use to pass form data to the server i.e., GET and POST. The method is specified with the method attribute to the form tag.

1.3.1 GET Method [April 16]

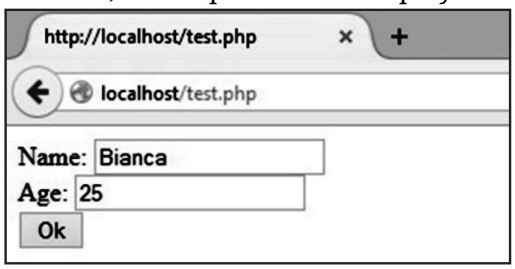
- The GET method sends the encoded user information appended to the page request (to the URL).
- The page and the encoded information are separated by the '?' character.

```
http://www.test.com/index.htm?name1=value1&name2=value2
```

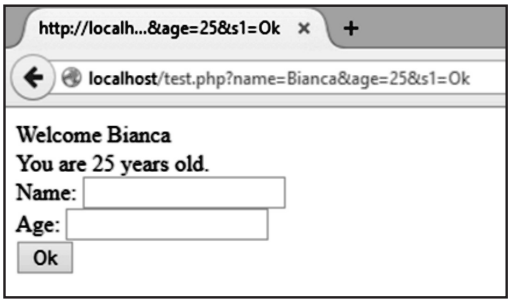
- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- The GET method is restricted to send up to 1024 characters only. Never use GET method if we have password or other sensitive information to be sent to the server.
- GET cannot be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY_STRING environment variable.
- The PHP provides $_GET associative array to access all the sent information using GET method.
- The following example puts the source code in test.php script:

```
<?php
   if( isset($_GET["s1"]) )
   {
      echo "Welcome ". $_GET['name']. "<br />";
      echo "You are ". $_GET['age']. " years old.";
   }
?>
<html>
   <body>
      <form action="<?php $_PHP_SELF?>" method="GET">
         Name: <input type="text" name="name" /><br>
         Age: <input type="text" name="age" /><br>
      <input type="submit" name="s1" value="Ok"/>
      </form>
   </body>
</html>
```

- The above program having two parts HTML and PHP. After executing the program in the browser, HTML part will be displayed first as follows as shown below:



- Enter name and age and submit the form. Now PHP will process the form and you will get the following output.



- After submitting the form the URL in the address bar is as follows:

```
http://localhost/test.php?name=Bianca&age=25&s1=Ok
```

It shows all the three form parameter's name/value pairs.

- In the program, the isset() method is used to check whether the "Ok" submit button is pressed or not. After entering name and age submitting the form will store the entered value of the text field into the $_GET array, and that can be displayed.

1.3.2 POST Method [April 16]

- The POST method transfers information via HTTP headers, not through the URL. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.
- The POST method does not have any restriction on data size to be sent. The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides $_POST associative array to access all the sent information using POST method.

- Try out following example by putting the source code in test.php script.

```
<?php
   if( isset($_POST["s1"]) )
   {
      echo "Welcome ". $_POST['name']. "<br />";
      echo "You are ". $_POST['age']. " years old.";
   }
?>
<html>
   <body>
      <form action="<?php $_PHP_SELF?>" method="POST">
         Name: <input type="text" name="name" /><br>
         Age: <input type="text" name="age" /><br>
         <input type="submit" name="s1" value="Ok"/>
      </form>
   </body>
</html>
```

Output:

- The output of the above program is same, except URL, the URL in the address bar will not get changed after submitting the form.

GET vs. POST Methods: **[April 16]**

- Both GET and POST create an array (For example, array(key => value, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.
- Both GET and POST are treated as $_GET and $_POST. These are superglobals, which means that they are always accessible, regardless of scope - and we can access them from any function, class or file without having to do anything special.
- The biggest difference between GET and POST requests is that, GET requests are idempotent i.e., one GET request for a particular URL, including form parameters, is the same as two or more requests for that URL.

- Thus, web browsers can cache the response pages for GET requests, because the response page doesn't change regardless of how many times the page is loaded.
- Because of idempotence, GET requests should be used only for queries such as splitting a word into smaller chunks or multiplying numbers, where the response page is never going to change.
- POST requests are not idempotent. This means that they cannot be cached, and the server is recontacted every time the page is displayed.
- We have probably seen the web browser prompt you with "Repost form data?" before displaying or reloading certain pages. This makes POST requests the appropriate choice for queries whose response pages may change over time—for example, displaying the contents of a shopping cart.

When to use GET?

- Information sent with the GET method is visible to everyone (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send.
- The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
- The GET may be used for sending non-sensitive data.

Note: GET should NEVER be used for sending passwords or other sensitive information!

When to use POST?

- Information sent with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send.
- Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.
- However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

1.3.3 Automatic Quoting of Parameters

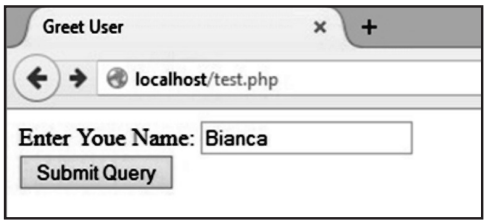
- The php.ini file is initialization file and is used to configure the behavior of PHP.
- In php.ini file if the option magic_quotes_gpc is enabled then PHP automatically add slashes (call addslashes()) on all cookie data and GET and POST parameters.
- **For Example:** If we enter the word "O'clock" as the form parameter value then the word is converted to "O\'clock".
- If we don't want to add slashes automatically and want to work with the strings as typed by the user, we can either disable magic_quotes_gpc in php.ini or use the stripslashes() function on the values in $_GET, $_POST, and $_COOKIES.

1.3.4 Self Processing Pages

- A single PHP program can be used to both generate a form and process it, using combination of HTML and PHP. This type of PHP page is known as self processing page.
- We can write a program which can decide whether to display a form or process it depending on the one of the parameter has been supplied or not.
- **For Example:**

```
<html>
   <head><title>Greet User</title></head>
   <body>
      <?php
         if(isset($_GET['name1']))
         {
            $nm = $_GET['name1'];
            echo "Hello " . $nm;
         } else {
      ?>
      <form action="<?php echo $_SERVER['PHP_SELF']?>" method="GET">
           Enter Your Name:
         <input type="text" name="name1" /> <br />
         <input type="submit" name="Ok" />
      </form>
      <?php
         }
      ?>
   </body>
</html>
```

- In this example if the value of the form parameter 'name1' is present then it gets processed, and in the absence of value only form will be displayed as shown below:

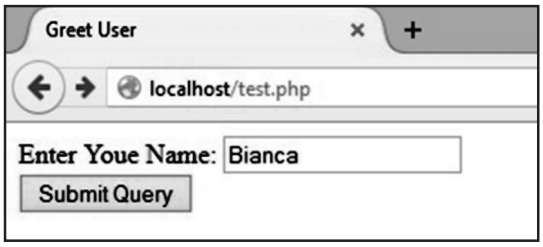


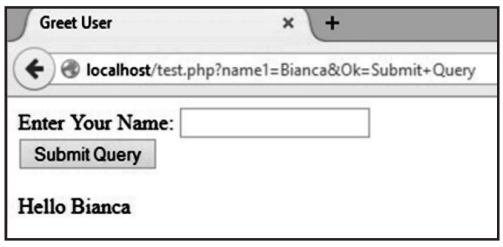
1.3.5 Sticky Forms [April 17, 19]

- Many web sites use a technique known as sticky forms. In sticky forms the values entered by user remain displayed with the form component, if we display the form after submit.
- **For example:** If we search Google (http://www.google.com) for "PHP Cookbook", the top of the results page contains another search box, which already contains "PHP Cookbook". To refine the search to "PHP Cookbook from O'Reilly", we can simply add the extra keywords.
- Consider the following Example:

```
<html>
   <head><title>Greet User</title></head>
   <body>
      <form action="<?php echo $_SERVER['PHP_SELF']?>" method="GET">
           Enter Your Name:
         <input type="text" name="name1" /> <br />
         <input type="submit" name="Ok" />
      </form>
      <?php
         if(isset($_GET['name1']))
         {
            $nm = $_GET['name1'];
            echo "Hello " . $nm;
         }
      ?>
   </body>
</html>
```

Output:



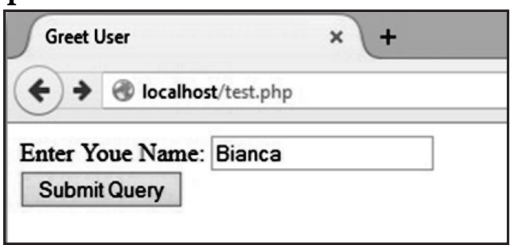


- The above form is not a sticky form, since the name entered in the text field is lost after submitting the form.
- To make form sticky we use the submitted form value as the default value when creating the HTML field. In the following program the submitted name is assigned to

the value of the 'name1' parameter. So that the value remain displayed in the text field after form submission.

```
<html>
   <head><title>Greet User</title></head>
   <body>
      <form action="<?php echo $_SERVER['PHP_SELF']?>" method="GET">
         Enter Youe Name:
         <input type="text" name="name1" value="<?php echo
                                                $_GET['name1'];?>"/> <br />
         <input type="submit" name="Ok" />
      </form>
      <?php
         if(isset($_GET['name1']))
         {
            $nm = $_GET['name1'];
            echo "Hello " . $nm;
         }
      ?>
   </body>
</html>
```

Output:

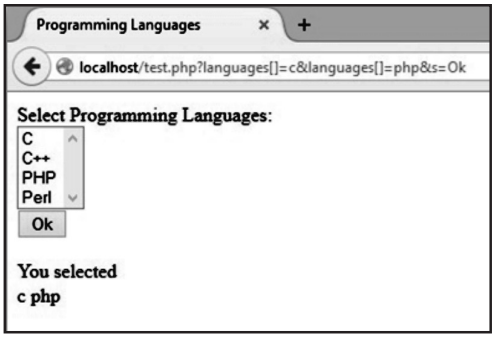


1.3.6 Multivalued Parameters [April 19]

- The HTML selection lists can allow multiple selections, when name of the field in the HTML form end with [] and the 'multiple' attribute is used.
- When the user submits the form, $_GET[' languages'] contains an array instead of a simple string. This array contains the values that were selected by the user.
- Following example shows multiple selections. The form provides the user with a set of programming languages name. When the user submits the form, he gets list of the selected of programming languages.

```
<html>
   <head>
      <title>Programming Languages</title>
   </head>
   <body>
      <form action="<?php echo $_SERVER['PHP_SELF']?>" method="GET">
         Select Programming Languages:<br />
         <select name="languages[]" multiple>
            <option value="c">C</option>
            <option value="c++">C++</option>
            <option value="php">PHP</option>
            <option value="perl">Perl</option>
         </select><br>
         <input type="submit" name="s" value="Ok" />
      </form>
      <?php
         if(isset($_GET['s']))
         {
            $lan = $_GET['languages'];
            echo "You selected<br>";
            foreach($lan as $k=>$v)
            echo "$v ";
         }
      ?>
   </body>
</html>
```

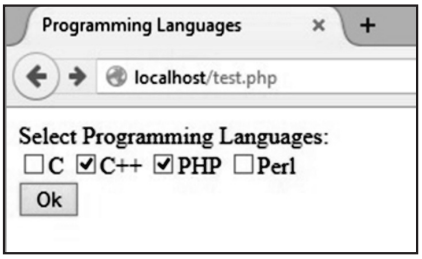
Output:



- The above program is rewritten using checkboxes in place of select box. Notice that only the HTML has changed, the code to process the form doesn't need to know whether the multiple values came from checkboxes or a select box.

```
<html>
   <head><title>Programming Languages</title></head>
   <body>
      <form action="<?php echo $_SERVER['PHP_SELF']?>" method="GET">
         Select Programming Languages:<br />
         <input type="checkbox" name="languages[]" value="c">C
         <input type="checkbox" name="languages[]" value="c++">C++
         <input type="checkbox" name="languages[]" value="php">PHP
         <input type="checkbox" name="languages[]" value="perl">Perl
         <br>
         <input type="submit" name="s" value="Ok" />
      </form>
      <?php
         if(isset($_GET['s']))
         {
            $lan = $_GET['languages'];
            echo "You selected<br>";
            foreach($lan as $k=>$v)
            echo "$v ";
         }
      ?>
   </body>
</html>
```

Output:



1.3.7 Sticky Multivalued Parameters

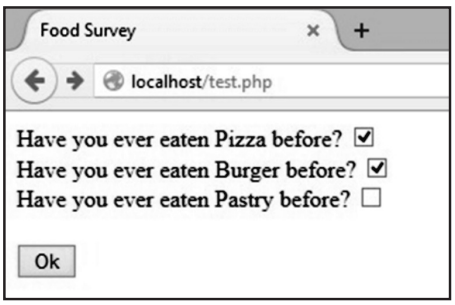
- A form can have multiple elements like text field, check boxes, radio button etc.; it is also possible to make that form sticky.

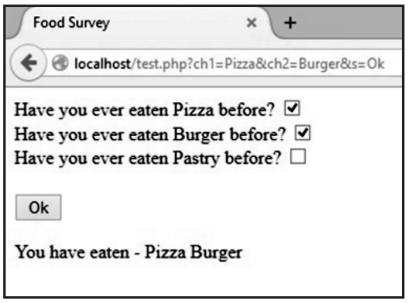
- It is also possible to make multiple selected form elements sticky. In such type of form each possible value in the form is one of the submitted values.
- In the program above the selected check boxes select (√) is lost after submitting the form.
- To make such type of form sticky so that the selected check boxes select (√) is not lost after submitting the form.

For Example:

```
<html>
   <head><title>Food Survey</title></head>
   <body>
      <form action="<?php echo $_SERVER['PHP_SELF']?>" method="GET">
         Have you ever eaten Pizza before?
      <input type="checkbox" name="ch1" value="Pizza" <?php
   if(isset($_GET['ch1']) and isset($_GET['s'])) echo "checked";?>><br />
         Have you ever eaten Burger before?
   <input type="checkbox" name="ch2" value="Burger" <?php
  if(isset($_GET['ch2']) and isset($_GET['s'])) echo "checked";?>><br />
         Have you ever eaten Pastry before?
   <input type="checkbox" name="ch3" value="Pastry" <?php
   if(isset($_GET['ch3']) and isset($_GET['s'])) echo "checked";?>><br />
   <br>
      <input type="submit" name="s" value="Ok" />
   </form>
   <?php
      if(isset($_GET['s']))
      {
         echo "You have eaten - ";
         if(isset($_GET['ch1'])) echo "$_GET[ch1] ";
         if(isset($_GET['ch2'])) echo "$_GET[ch2] ";
         if(isset($_GET['ch3'])) echo "$_GET[ch3]";
      }
   ?>
   </body>
</html>
```

Output:





1.3.8 File Uploads [April 16, 19, Oct. 18]

- In PHP to handle file uploads, $_FILES array is used. The elements of the $_FILES array gives information about the uploaded file.
- The keys are:
  1. **name:** The name of the file, as supplied by the browser.
     `$_FILE['filename']['name']`
  2. **Type:** The type of the uploaded file. For example, image/jpeg.
     `$_FILE['filename']['type']`
  3. **Size:** The size of the uploaded file (in bytes). If the user attempted to upload a file that was too large, the size is reported as 0.
     `$_FILE['filename']['size']`
  4. **tmp_name:** The name of the temporary file on the server that holds the uploaded file. If the user attempted to upload a file that was too large, the name is reported as "none".
     `$_FILE['filename']['tmp_name']`
  5. **Error**: The error code resulting from the uploaded file.
     `$_FILE['filename']['error']`
- To test whether a file was successfully uploaded or not use the function is_uploaded_file(), as follows:

```
if (is_uploaded_file($_FILES['filename']['tmp_name'])
{
    // successfully uploaded
}
```

- Files are stored in the server's default temporary files directory, which is specified in php.ini with the upload_tmp_dir option.
- We can also move the file to a permanent location using move_uploaded_file() function.

```
// move the file: move_uploaded_file() also does a check of the file's
// legitimacy, so there's no need to also call is_uploaded_file()
move_uploaded_file($_FILES['filename']['tmp_name'],
                                                  '/path/to/file.txt');
```

- The value stored in tmp_name is the complete path to the file, not just the base name. Use basename( ) to chop off the leading directories if needed.
- Second parameter of move_uploaded_file()is the target path where the file is moved and stored permanentaly after upload.

Example:

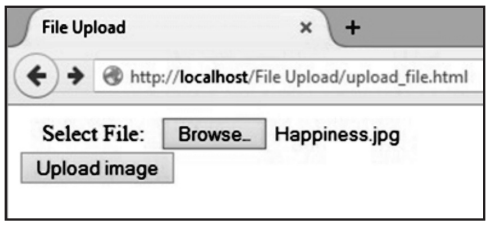
upload.html file:

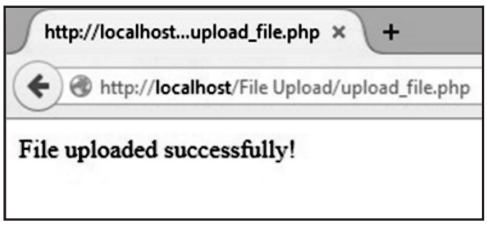
```
<html>
   <head><title>File Upload</title></head>
   <body>
      <form action="upload.php" method="post" enctype="multipart/form-data">
         Select File:
         <input type="file" name="fileToUpload"/>
         <input type="submit" value="Upload Image" name="submit"/>
      </form>
   </body>
</html>
```

upload.php file:

```
<?php
   $target_path = "E:/";
   $target_path = $target_path.basename
                                        ($_FILES['fileToUpload']['name']);
   if(move_uploaded_file($_FILES['fileToUpload']
                                                  ['tmp_name'], $target_path))
   {
      echo "File uploaded successfully!";
   } else {
      echo "Sorry, file not uploaded, please try again!";
   }
?>
```

Output:





- In the above program first, you select a file using Browse button then click on 'Upload image' button, so next page will be displayed showing the message "File uploaded successfully!". The uploaded file can be seen in the E:/ directive.

1.3.9 Validating Forms

- When user input data in a form it should be validated before storing it. There are two ways to validate data i.e., client side validation and server side validation.
- For client side validation JavaScript is used, which is not secure because user can turn off JavaScript or browser may not support it.
- The more secure way is to use PHP for the validation. PHP is used for server side validation.

Example:

form_validation.html

```
<html>
   <head><title>Validated Form</title></head>
   <body>
      <form name="form1" method="POST" action="form_validation.php">
            Your Name (Required):
         <input name="name" type="text">
         <br>
            Username (at least 5 characters):
         <input name="username" type="text">
         <br>
         <input name="send" type="submit" value="Submit">
      </form>
   </body>
</html>
```

form_validation.php

```
<?php
   // First check if the form button has been submitted
   if(isset($_POST['send'])){
      if(!empty($_POST['name'])){ // Check if the name is empty
         echo "Name: " . $_POST['name'] . "<br />";
      }else{
         echo "Name is missing.";
         echo "<br />";
      }
```

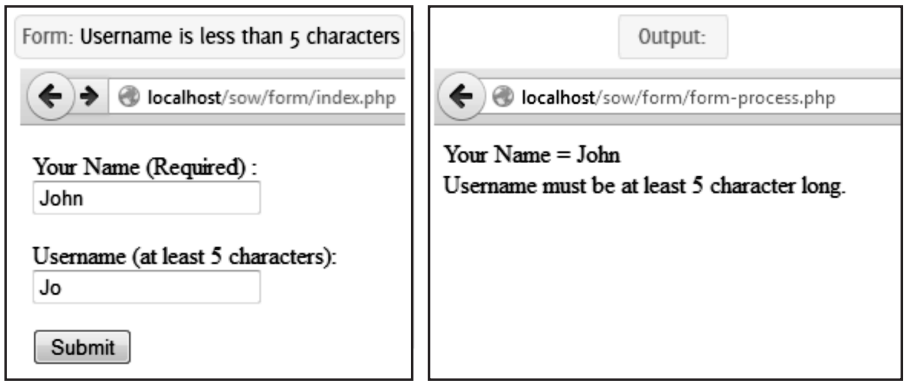
```
        if(!empty ($_POST['username'])){// Check if the username is empty
           if(strlen($_POST['username']) < 5 ){ // Check
                                    if the username is at least 5 characters.
              echo "Username must be at least 5 characters long.";
              echo "<br />";
           }else{
              echo "User Name: " . $_POST['username'];
           }
        }else{
              echo "Username is missing.";
              echo "<br />";
        }
     }else{
        echo "Unauthorized access to this page.";
     }
  ?>
```

- In the above program there are two pages: HTML form to accept name and username from user and, second page is PHP form processing page, is used to validate user's data.
- Any user may come to this page directly. To prevent unauthorized access, we will check whether the user comes to this page clicking the form button the following condition is used:

  ```
  if(isset($_POST['send']))
  ```

- If the user comes here directly the function will return false and the message "Unauthorized access to this page." is displayed.
- PHP has another function named empty() that can check whether a variable is empty. We will check whether the name field("name") in the form is empty.
- If the user has entered his name that means the field is not empty and in this case the function empty() will return true, and the name prints.
- If the user didn't enter his name, the function will return false and the else statement will be executed. Then, the message "Name is missing." will be printed.

- In the same way, the username field is also checked using empty() function. The strlen() function check whether "username" is less than 5 characters. If it is, the warning "Username must be at least 5 characters long." will be displayed.



- It is easy for hackers to inject malicious code through form inputs, remove important files from server, damage your database etc. So, you need to sanitize user inputs to remove suspicious characters or to alter user inputs to usable form.
- There are few built-in php functions that help to sanitize form data. e.g. strip_tags() to remove any HTML and PHP tags from a string, htmlspecialchars(), and htmlentities() to converts all applicable characters to HTML entities etc.

For example:

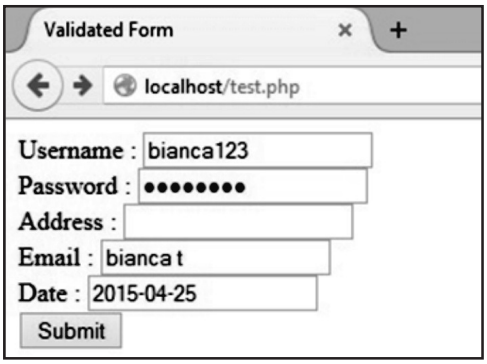
```
if ($_SERVER["REQUEST_METHOD"] == "POST")
{
   $name = test_input($_POST["name"]);
   $email = test_input($_POST["email"]);
   $website = test_input($_POST["website"]);
   $comment = test_input($_POST["comment"]);
   $gender = test_input($_POST["gender"]);
}
function test_input($data) {
   $data = trim($data);
   $data = stripslashes($data);
   $data = htmlspecialchars($data);
   return $data;
}
<form method="post" action="<?php echo
                                htmlspecialchars($_SERVER["PHP_SELF"]);?>">
----
</form>
```

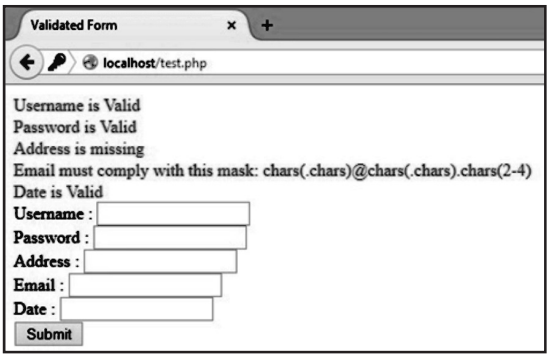
- Regular expression can be used to validate form input data. The following example shows how to validate a form using regular expression.

```
<?php
   echo "<font color=#FF0000>";
   if(isset($_POST['send']))
   {
      if(!empty($_POST['username']))
      {
         if(preg_match("/^[A-Za-z0-9]{3,20}$/",$_POST['username']))
            echo "Username is Valid<br>";
         else echo "Username is invalid<br>";
      }
   else  echo "Username is missing<br>";
   if(!empty($_POST['pass']))
   {
      if(preg_match("/^.*(?=.{8,})(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z]).*$/",
                                                              $_POST['pass']))
         echo "Password is Valid<br>";
      else echo "Password must be at least 8 characters and must contain
         at least one lower case letter, one upper case
                                                  letter and one digit<br>";
   }
   else  echo "Password is missing<br>";
   if(!empty($_POST['address']))
   {
      if(preg_match("/^[a-zA-Z0-9 -.,:\"\']+$/", $_POST['address']))
      echo "Address is Valid<br>";
      else echo "Address must be only letters,
                        numbers or one of the following - . ,: \" \'<br>";
   }
   else echo "Address is missing<br>";
   if(!empty($_POST['email']))
   {
      if(preg_match("/^[a-zA-Z]\w+(\.\w+)*\@\w+(\.[0-9a-zA-Z]+)
                                     *\.[a-zA-Z]{2,4}$/", $_POST['email']))
         echo "Email is Valid<br>";
      else echo "Email must comply with this mask:
         chars(.chars)@chars(.chars).chars(2-4)<br>";
   }
   else  echo "Email is missing<br>";
```

```
    if(!empty($_POST['date']))
    {
        if(preg_match("/^[0-9]{4}-[0-9]{1,2}-[0-9]{1,2}$/", $_POST['date']))
            echo "Date is Valid<br>";
        else echo "Date must comply with this mask: YYYY-MM-DD<br>";
    }
    else  echo "Date is missing<br>";
}
    echo "</font>";
?>
<html>
    <head><title>Validated Form</title></head>
    <body>
        <form name="form1" method="POST" action="test.php">
            Username:
        <input name="username" type="text">  <br>
            Password:
        <input name="pass" type="password">  <br>
            Address:
        <input name="address" type="text">  <br>
            Email:
        <input name="email" type="text">  <br>
            Date:
        <input name="date" type="text">  <br>
        <input name="send" type="submit" value="Submit">
        </form>
    </body>
</html>
```

Output:





1.4 SETTING RESPONSE HEADERS [April 17]

- We have already seen that when a client send a request to server for a web page, is called as HTTP request.
- The server sends back a response to the client is called as HTTP response. This HTTP response contains headers which include some information like, the type of content in the body of the response, how many bytes are in the body, when the response was sent, etc.
- If server sends back HTML then PHP and Apache take care of the headers, like, identifying the document as HTML, calculating the length of the HTML page, and so on.
- If we want to send back something that is not HTML, like, set the expiration time for a page, redirect the client's browser or generate a specific HTTP error, in that case we need to use the header() function.
- The header() function sends a raw HTTP header to a client.
- **Syntax:** `header(string, replace, http_response_code)`

  where, string: Specified header string to send.

  replace: Optional, indicates whether the header should replace previous or add second header. Default is TRUE (will replace). FALSE allow multiple headers of same type.

  http_response_code: Optional, force the HTTP response code to the specified value.
- We must set the headers before any o the body is generated. This means that all calls to header()must happen at the very top of your file, even before the <html> tag.
- **For example:**

```
<?php
   header('Content-Type: text/plain');
?>
<html>
   . . . .
</html>
```

- Attempting to set headers after the document has started gives this warning:

```
Warning:  Cannot add header information - headers already sent
```

Different Content Types:

- Content-Type header identifies the type of document being returned. Content-type ensures that the browser displays the content correctly.
- Ordinarily this is "text/html", indicating an HTML document is about to receive by the browser.

```
header("Content-type: text/html");
```

- Other useful document type is header(“Content-type: text/plain"); forces the browser to treat the page as plain text.
- Other content types are:

```
header(“Content-type: image/jpeg");
```

send a header to the web browser telling it the image type of data that it is is about to receive.

```
header(“Content-type: application/pdf");
```

will be outputting a PDF.

Redirections:

- Redirection means to send the user to a new URL. For this the Location header is used with the header function:

```
header('Location: http://www.nirali.com/books.html');
```

- When this statement executes the page containing the above statement is redirected to the page /index.html.
- If we want to pass variables to the new page, you can include them in the query string of the URL:

```
header('Location: http://www.nirali.com/books.php?id=12');
```

- The URL that you are redirecting a user to is retrieved with GET. You can't redirect someone to retrieve a URL via POST.

Expiration:

- A server can explicitly inform the browser, a specific date and time for the document to expire. The server can also inform expiration date and time to proxy cache.
- To set the expiration time of a document, use the Expires header:

```
header('Expires: Fri, 18 Jan 2014 05:30:00 GMT');
```

- Following example shows how a document expires three hours from the time the page was generated:

```
$now = time( );
$then = gmstrftime("%a, %d %b %Y %H:%M:%S GMT", $now + 60*60*3);
header("Expires: $then");
```

- The function time() returns current time and function gmstrftime() is used to generate expiration date string.

Authentication:

- HTTP authentication works through request headers and response statuses.
- A browser can send a username and password in the request headers. If the username and password aren't sent or aren't satisfactory, the server sends a "401 Unauthorized" response and identifies a string (realm) via WWW-Authenticate header. This typically pops up an "Enter username and password for ..." dialog box on the browser, and the page is then re-requested with the updated username and password in the header.

- The $_SERVER['PHP_AUTH_USER'] and $_SERVER['PHP_AUTH_PW'] global variables contain the username and password supplied by the user, if any.
- To deny access to a page, send a WWW-Authenticate header identifying the authentication realm as part of a response with status code 401:

```
header('WWW-Authenticate: Basic realm="My Website"');
header('HTTP/1.0 401 Unauthorized');
   echo "You need to enter a valid username and password.";
exit;
```

- The header() function is used to send a message "Authentication required" to the client browser causing it to pop up a username/password input window.
- Once, the user has entered username and password, the same URL will be called again with the predefined variables PHP_AUTH_USER and PHP_AUTH_PW set with username and password respectively.

For example:

```
<?php
   if(!isset($_SERVER['PHP_AUTH_USER']))
   {
      header('WWW-Authenticate: Basic realm="My Website"');
      header('HTTP/1.0 401 Unauthorized');
         echo "You need to enter a valid username and password.";
      exit;
   }
   else
   {
      echo "Hello ". $_SERVER['PHP_AUTH_USER'] . "<br>";
      echo "You entered". $_SERVER['PHP_AUTH_PW'] . "as your password";
   }
?>
```

1.5 MAINTAINING STATE [Oct. 17]

- As we provide input to some application via the user interface, we change the state of the application.
- State is the exact condition, movement to movement, of all the data and variables in the application.
- We know that HTTP is a stateless protocol i.e., HTTP supports only one request per connection. This means that with HTTP the client connects to the server to send one request and then disconnects.

- In other words, there is no way for a server to recognize that a sequence of requests all originate from the same client.
- When we programming for Web, you cannot keep track of a sequence of requests from a single user, because HTTP is stateless.
- For example, in a shopping-cart application we need to know that the same user adds items in his cart or removes items from his cart, every time when he sends request to add or remove items.
- So building an online application with PHP requires you to use some mechanism for maintaining state.
- PHP is able to bridge the gap between each request-response and provides persistence of data and prolong interactivity.
- To get around the Web's lack of state, software programmers have come up with many tricks to keep track of state information between requests which is also known as session tracking.
- Some techniques are:

1. **Hidden Fields:** One session tracking technique is to use hidden form fields to pass data from one page to another. PHP treats hidden form fields just like normal form fields, so the values are available in the $_GET or $_POST arrays. So after submitting a form, the form data can be fetched by the next page from $_GET or $_POST arrays.

   However, a more common and popular technique is to assign each user a unique identifier using a single hidden form field.

   For example, consider a form where a user selects a product from a drop-down box. We can simply place the product ID into a hidden form field.

```
<form action="myform.php" method="post">
   <select name="selected_product_id">
      <option value="121">Product 121</option>
      <option value="122">Product 122</option>
   </select>
   <input type="submit" value="Select Product">
</form>
```

   We could return a page to him using the following code:

```
<input type="hidden" name="chosen_product_id" value="<?php echo
                                        $_POST[selected_product_id];?>">
```

   The chosen_product_id value can be fetched after submitting the form.

   While hidden form fields work in all browsers, they work only for a sequence of dynamically generated forms, so they aren't as generally useful as some other techniques.

2. **URL Rewriting:** In this technique you can add some extra information in the URL. For example, you can assign a unique ID to a user as given below, so that this ID can be passed to that page.

   ```
   http://www.nirali.com/catalog.php?userid=123
   ```

   This method is very insecure because everyone can see the values. User can also change the value of variable in the URL.

3. **Cookies:** A third technique for maintaining state with PHP is to use cookies. A cookie is a bit of information that the server can give to a client. On every subsequent request the client will give that information back to the server, thus identifying itself. Cookies are useful for retaining information through repeated visits by a browser. **[April 16]**

- The best way to maintain state with PHP is to use the built-in session-tracking system. This system lets us to create persistent variables that are accessible from different pages of the application, as well as in different visits to the site by the same user.

1.5.1 Cookies **[April 16, Oct. 17, 18]**

- A cookie is a small piece of data in the form of a name-value pair that is sent by a Web server and stored by the browser on the client machine.
- This information is used by the application running on the server side to customize the web page according to the past history of transactions from that client machine.
- A server can send one or more cookies to a browser in the headers of a response. Some of the cookie's fields indicate the pages for which the browser should send the cookie as part of the request.
- Servers can store any data they like there in the value field of the cookie (within limits), such as a unique code identifying the user, preferences, etc.
- PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.
- Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.

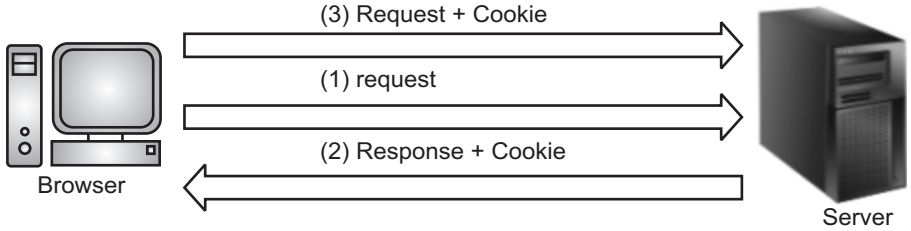


Fig. 1.1: Cookies

- In short, cookie can be created, sent and received at server end.

- Use the setcookie() function to send a cookie to the browser. Like header(), setcookie() must be called before any HTML is printed to the client browser because cookies are set in the HTTP headers, which must be sent before any HTML.
- The **syntax** is,

```
setcookie(name [, value [, expire [, path [, domain [, secure ]]]]]);
```

Here, is the detail of all the arguments in above syntax:

- **Name:** Name of the cookie, stored in an environment variable called $_COOKIE. This variable is used while accessing cookies.
- **Value:** Value of the named variable.
- **Expiry:** A UNIX timestamp denoting the time at which the cookie will expire.
- **Path:** This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
- **Domain:** The browser will return the cookie only for URLs within this domain. The default is the server hostname.
- **Secure:** This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

- When a browser sends a cookie back to the server, you can access that cookie through the $_COOKIE array. The key is the cookie name, and the value is the cookie's value field.
- For example, to set cookie in one page:

```
<?php
   setcookie("ID", "121");
   setcookie("name", "Laptop");
?>
```

- Can be access in another page as follows:

```
<?php
   if( isset($_COOKIE["ID"]))
   {
      echo "Product ID: " . $_COOKIE["ID"] . "<br />";
      echo "Product Name: " . $_COOKIE["name"];
   }
   else
      echo "Sorry... Not recognized" . "<br />";
?>
```

Output:

```
Product ID: 121
Product Name: Laptop
```

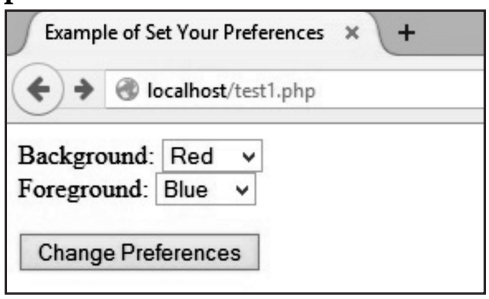
- For instance, the following code at the top of a page keeps track of the number of times the page has been accessed by this client:

```
<?php
   if(!isset($_COOKIE['accesses']))
      setcookie('accesses', 0);
   else{
      $page_accesses = $_COOKIE['accesses'];
      setcookie('accesses', ++$page_accesses);
         echo "This page is accessed " . $_COOKIE['accesses']
                                                        . " times";
   }
```

- Initially the cookie named “accesses” is not set, so it is set to 0. After that every time the page is accessed the program displays how many times the page is accessed.
- Following example shows an HTML page that gives a range of options for background and foreground colors.

```
<html>
   <head>
      <title>Example of Set Your Preferences</title>
   </head>
   <body>
      <form action="prefsexample.php" method="post">
            Background:
         <select name="background">
            <option value="blue">Blue</option>
            <option value="black">Black</option>
            <option value="white">White</option>
            <option value="red">Red</option>
         </select><br />
            Foreground:
         <select name="foreground">
            <option value="blue">Blue</option>
            <option value="black">Black</option>
            <option value="white">White</option>
            <option value="red">Red</option>
         </select><p />
         <input type="submit" value="Change Preferences">
      </form>
   </body>
</html>
```

Output:



- The form in above Example submits to the PHP script prefs.php, which is shown in following Example.
- This script sets cookies for the color preferences specified in the form. Note that the calls to setcookie() are made before the HTML page is started.

```
<?php
   $colors = array('black' => '#000000',
             'white' => '#ffffff',
             'red'   => '#ff0000',
             'blue'  => '#0000ff');
   $bg_name = $_POST['background'];
   $fg_name = $_POST['foreground'];
   setcookie('bg', $colors[$bg_name]);
   setcookie('fg', $colors[$fg_name]);
?>
<html>
   <head><title>Example of Preferences Set</title></head>
   <body>
      Thank you. Your preferences have been changed to:<br />
      Background: <?= $bg_name?><br />
      Foreground: <?= $fg_name?><br />
      Click <a href="prefs-demo.php">here</a>
                                          to see the preferences in action.
   </body>
</html>
```

- The page created by above Example contains a link to another page, shown in following Example, that uses the color preferences by accessing the $_COOKIE array.

```
<html>
   <head>
      <title>Example of Front Door</title>
   </head>
   <?php
      $bg = $_COOKIE['bg'];
      $fg = $_COOKIE['fg'];
   ?>
   <body bgcolor="<?= $bg?>" text="<?= $fg?>">
      <h1>Welcome to the Store</h1>
```

- We have many fine Services for you to view. Please feel free to browse the aisles and stop an assistant at any time. But remember, you break it you bought it!<p>
- Would you like to change your preferences?

```
   </body>
</html>
```

Deleting Cookie with PHP:

- Officially, to delete a cookie you should call setcookie() with the name argument only but this does not always work well, however, and should not be relied on.
- It is safest to set the cookie with a date that has already expired:

```
<?php
   setcookie( "name", "", time()- 60, "/","", 0);
   setcookie( "age", "", time()- 60, "/","", 0);
?>
<html>
   <head>
      <title>Deleting Cookies with PHP</title>
   </head>
   <body>
      <?php echo "Deleted Cookies"?>
   </body>
</html>
```

1.5.2 Sessions [April 17]

- A session can be defined as "a series of related interactions between a single client and the web server, which takes place over a period of time".

- Sessions allow you to store data in the web server that associated with a session ID. Once you create a session, PHP sends a cookie that contains the session ID to the web browser.
- In the subsequent requests, the web browser sends the session ID cookie back to the web server so that PHP can retrieve the data based on the session ID and make the data available in your script.
- Session provides a way to preserve certain data across subsequent accesses. This enables us to build more customized applications.
- Sessions allow we to easily create multipage forms like shopping carts, save user authentication information from page to page, and store persistent user preferences on a site.
- PHP session is used to store and pass information from one page to another temporarily (until user close the website).
- An overview of PHP session management is shown in Fig. 1.2.

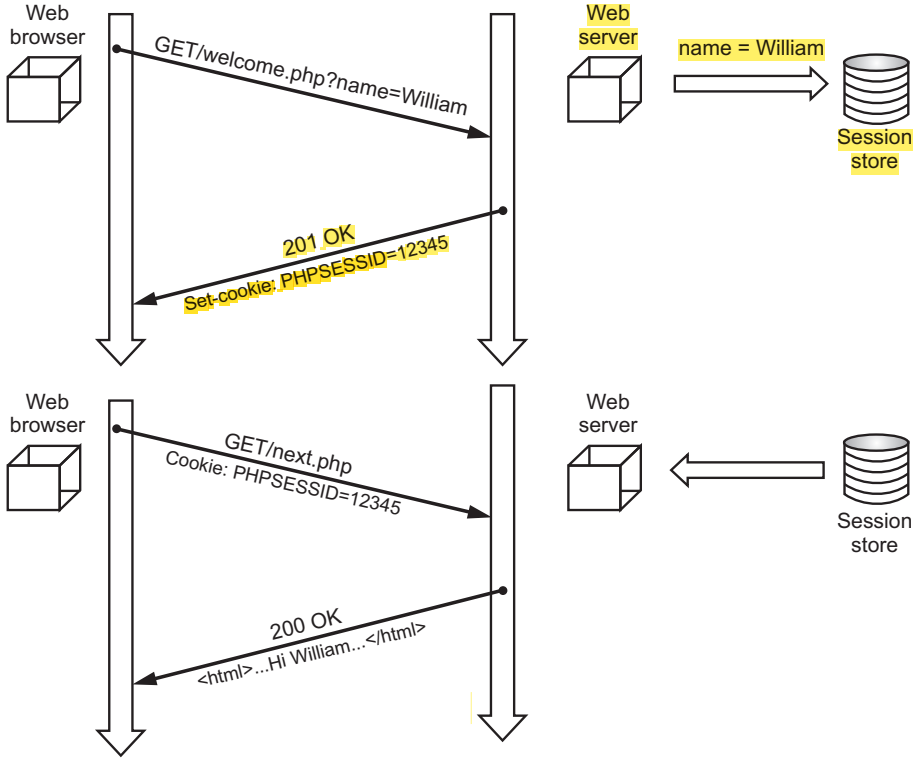


Fig. 1.2: The interaction between the browser and the server when initial requests are made to session-based application

- When user first enters the session-based application by making a request to a page that starts a session, PHP generates a session and creates a file that stores the session-related variables.

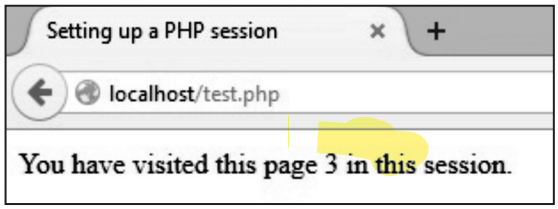
- PHP sets a cookie named PHPSESSID to hold the session ID in the response the script generates. If the user's browser does not support cookies or has cookies turned off, the session ID is added in URLs within the web site. The browser then records the cookie and includes it in subsequent requests.
- In the example shown above, the script welcome.php records session variables in the session store, and a request to next.php then has access to those variables because of the session ID.
- A session ends when the user closes the browser or after leaving the site, the server will terminate the session after a predetermined period of time.
- PHP session_start() function is used to start the session. It starts a new or resumes existing session. It returns existing session if session is created already. If session is not available, it creates and returns new session.

  Syntax: `bool session_start (void)`
- $_SESSION is an associative array that contains all session variables. It is used to set and get session variable values.
- Session variables are stored in associative array called $_SESSION[]. These variables can be accessed during lifetime of a session.
- The following example starts a session then registers a variable called counter that is incremented each time the page is visited during the session.
- Make use of isset() function to check if session variable is already set or not. Put this code in a test.php file and load this file many times to see the result:

```
<?php
   session_start();
   if(isset( $_SESSION['counter']))
   {
      $_SESSION['counter'] += 1;
   }
   else
   {
      $_SESSION['counter'] = 1;
   }
      $msg = "You have visited this page ".  $_SESSION['counter']
                                          . " in this session.";
?>
<html>
   <head>
      <title>Setting up a PHP session</title>
```

```
    </head>
    <body>
        <?php  echo ( $msg );?>
    </body>
</html>
```

Output:



- We can register a variable with the session by passing the name of the variable to session_register(). If register_globals is enabled in the php.ini file, the variables are also set directly. Because the array and the variable both reference the same value, setting the value of one also changes the value of the other.
- We can unregister a variable from a session, which removes it from the data store, by calling session_unregister().

How to Access Data in the Session?

- Unlike a cookie, we can store any types of data in sessions. You store data as keys and values in the $SESSION[] superglobal array.
- For example, in the index.php file, we store user string and roles array in the session as follows:

```
<?php
    session_start();
    $_SESSION['user'] = "admin";
    $roles = array("admin", "approver", "editor");
    $_SESSION['roles'] = $roles;
    session_write_close();
?>
    <a href="test1.php">Go to profile page</a>
```

- In the profile.php file, you can access session data as follows:

```
<?php
    session_start();
    if(isset($_SESSION['user']))
    {
        echo sprintf("Welcome %s!<br>",$_SESSION['user']);
    }
```

```
        if(isset($_SESSION['roles']))
        {
           echo sprintf('Your roles: %s', implode($_SESSION['roles'],','));
        }
?>
```

Output:

```
Welcome admin!
Your roles: admin,approver,editor
```

Destroy a PHP Session: **[April 14]**

- To remove all global session variables and destroy the session, use session_unset() and session_destroy() functions.
- A PHP session can be destroyed by session_destroy() function. This function does not need any argument and a single call can destroy all the session variables.
- If you want to destroy a single session variable then you can use unset() function to unset a session variable.
- Here, is the example to unset a single variable:

```
<?php
   unset($_SESSION['counter']);
?>
```

- Here, is the call which will destroy all the session variables:

```
<?php
   session_start();
?>
<html>
   <body>
      <?php
         // remove all session variables session_unset();
         // destroy the session session_destroy();
         echo "All session variables are now removed
                                              and the session is destroyed."
      ?>
   </body>
</html>
```

Alternatives to Cookies:

- By default, the session ID is passed from page to page using PHPSESSID cookie. The session ID can also be passed using form fields and URLs.
- Passing the session ID via hidden form fields is extremely awkward, as it forces you to make every link between pages be a form's submit button.

- The URL system for passing around the session ID, however, is very elegant. PHP can rewrite your HTML files, adding the session ID to every relative link. For this to work, PHP must be configured with the -enable-trans-id option when compiled.
- But the performance is decreased, because PHP must parse and rewrite every page. Busy sites must be used cookies, as cookies do not slowdown the website caused by page rewriting.

Custom Storage:

- By default, the session information is stored in files in server's temporary directory.
- Each session's variables are stored in the separate file. The location of the session files can be changed with session Save_path in PHP. ini file.
- The session information is stored in two formats.
  1. PHP's built_in format.
  2. WDDX.
- Here, the session data in a database which will be shared between multiple sites. Here different functions are required for opening new session, closing a session, reading session information, writing session information, destroying a session etc.
- To use the custom session store use the following code:

```
<Directory "/var/html/sample">
   php_value session.save_handler user
   php_value session.save.path mydb
   php_value session.name session_store
</Directory>
```

Combining Cookies and Sessions:

- Using a combination of cookies and session, you can preserve state across visits. When a user leaves the site that state should be forgotten.
- Any state that should persist between user visits, such as a unique user ID, can be stored in a cookie.
- With the user's ID, we can retrieve the user's more permanent state, such as display preferences, mailing address, and so on, from a permanent store, such as a database.
- Following example allows the user to select text and background colors and stores those values in a cookie. Any visits to the page within the next week send the color values in the cookie.

```
<?php
   if($_POST['bgcolor'])
   {
      setcookie('bgcolor', $_POST['bgcolor'], time()
                                          + (60 * 60 * 24 * 7));
   }
      $bgcolor = empty($bgcolor)? 'black': $bgcolor;
?>
```

```
<body bgcolor="<?= $bgcolor?>">
   <form action="<?= $PHP_SELF?>" method="POST">
      <select name="bgcolor">
         <option value="black">Black</option>
         <option value="white">White</option>
         <option value="gray">Gray</option>
         <option value="blue">Blue</option>
         <option value="green">Green</option>
         <option value="red">Red</option>
      </select>
      <input type="submit" />
   </form>
</body>
```

Example:

- A PHP script to create a login form with user name and password, once the user login, the second form should be display to accept the user details (roll_no, name, city).
- If the user does not enter information within specified time limit, expire his session and give the warning.

login.html File:

```
<html>
   <head><title>login form </title></head>
   <body>
      <form method = "POST" action = "login.php">
            Username :
         <input type = "text" name = "user"><br>
            Password :
         <input type = "password" name = "pass"><br><br>
         <input type = "submit" value="Submit">
      </form>
   </body>
</html>
```

login.php File:

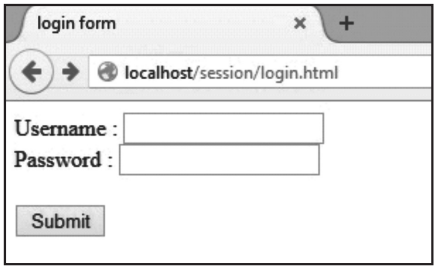
```
<?php
   session_start();
   $_SESSION["tm"] = time();
?>
```

```
<form method = "GET" action = "new.php">
      Roll No : <input type = "text" name = "rno"><br>
      Name : <input type = "text" name = "nm"><br>
      City : <input type = "text" name = "ct"><br><br>
   <input type = "submit" value="Submit">
</form>
```

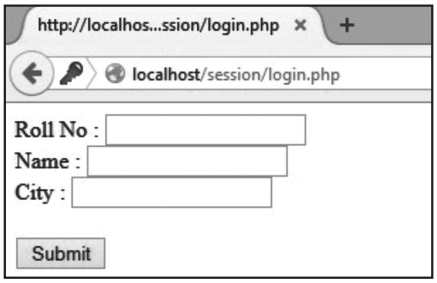
new.php File:

```
<?php
   session_start();
   $newt=$_SESSION['tm'] + 20;
   if($newt < time())
      echo "Time out";
   else
   {
      echo "Roll No = $_GET[rno] <br>";
      echo "Name = $_GET[nm] <br>";
      echo "City = $_GET[ct]";
   }
   session_destroy( );
?>
```

Output: When we run login.html file, the following screen will be displayed which will ask us to enter username and password.



After we enter any username and password, the following screen will be display:



If we fail to enter all the details within 20 second, the following screen will be displayed:

If we enter all the details within 20 second, the following screen will be displayed:

- In login.php file the session 'tm' is set with the current time using time(). When we submit this form after entering all the fields, control goes to new.php file, where session variable 'tm', which now stores the form submission time, is incremented by 20. This total value compared with the current time so that you can find out whether the form is submitted within 20 second or not.

1.6 PHP ERROR HANDLING

- Error handling is the important part of any program. While creating PHP scripts the errors need to be handled. If your code lacks of error checking, your program may be open to security risks.
- Error handling is the process of catching errors raised by your program and then taking appropriate action.
- It is very simple in PHP to handle errors. An error message with filename, line number and a message describing the error is sent to the browser.
- The first example shows a simple script that opens a text file:

```
<?php
   $file=fopen("pragati.txt","r");
?>
```

- If the file does not exist you might get an error message.
- While writing your PHP program you should check all possible error condition and take appropriate action when required.

```
<?php
   if(!file_exists("pragati.txt"))
   {
      die("File not found");
   }
```

```
    else
    {
        $file=fopen("pragati.txt","r");
    }
?>
```

- Now if the file does not exist we get an error message “File not found” but if the file exists then the file will get open.
- In the above program the error handling mechanism is used, hence, this code is more efficient than the earlier code. The first code is not the right way.
- In the following section some alternative PHP error handler functions are described.

1.6.1 Creating a Custom Error Handler

- In PHP we can define your own error handler function to handle any error.
- Creating a custom error handler is quite simple. We simply create a special function that can be called when an error occurs in PHP.
- To create custom error handler first use the library function set_error_handler(). This function sets a user-defined error handler function.

  Syntax:

```
mixed set_error_handler (callable $error_handler[,
                         int $error_types = E_ALL | E_STRICT])
```

  Here, $error_handler contains name of the user defined function, that is used to handle error. Second parameter $error_types is optional.
- The **syntax** of the user defined function is:

```
bool handler (int $errno , string $errstr [, string $errfile
                                  [, int $errline [, array $errcontext ]]])
```

- The parameters are described as follows:

Parameter	Description
error_level	Required: Specifies the error report level for the user-defined error. Must be a value number.
error_message	Required: Specifies the error message for the user-defined error.
error_file	Optional: Specifies the file name in which the error occurred.
error_line	Optional: Specifies the line number in which the error occurred.
error_context	Optional: Specifies an array containing every variable and their values in use when the error occurred.

Error Report Levels:

- These error report levels are the different types of error the user-defined error handler can be used for:

Value	Constant	Description
1	E_ERROR	Fatal run-time errors. Execution of the script is halted.
2	E_WARNING	Non-fatal run-time errors. Execution of the script is not halted.
4	E_PARSE	Compile-time parses errors. Parse errors should only be generated by the parser.
8	E_NOTICE	Run-time notices. The script found something that might be an error, but could also happen when running a script normally.
16	E_CORE_ERROR	Fatal errors that occur during PHP's initial startup.
32	E_CORE_WARNING	Non-fatal run-time errors. This occurs during PHP's initial startup.
256	E_USER_ERROR	Fatal user-generated error. This is like an E_ERROR set by the programmer using the PHP function trigger_error().
512	E_USER_WARNING	Non-fatal user-generated warning. This is like an E_WARNING set by the programmer using the PHP function trigger_error().
1024	E_USER_NOTICE	User-generated notice. This is like an E_NOTICE set by the programmer using the PHP function trigger_error().
2048	E_STRICT	Run-time notices. Enable to have PHP suggest changes to your code which will ensure the best interoperability and forward compatibility of your code.
4096	E_RECOVERABLE_ERROR	Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle.
8191	E_ALL	All errors and warnings, except level E_STRICT (E_STRICT will be part of E_ALL as of PHP 6.0).

Example: Creating a function to handle errors.

```
<?php
   function on_error($num, $str, $file, $line)
   {
      print "Encountered error $num in $file, line $line: $str\n";
   }
      set_error_handler("on_error");
      print $a;
?>
```

Output:

```
Encountered error 8 in C:\wamp\www\test.php, line 7: Undefined variable: a
```

- The above program first call the function set_error_handler(). Then the set_error_handler() function call on_error() function which display the message with error number, file name, line number and the error message.

1.6.2 Trigger an Error

- In a script where users can input data it is useful to trigger errors when an illegal input occurs. In PHP, this is done by the trigger_error() function.
- In following example an error occurs if the "test" variable is bigger than "1".

```
<?php
   $test=2;
   if ($test>1)
   {
      trigger_error("Value must be 1 or below");
   }
?>
```

Output:

```
Notice: Value must be 1 or below
in C:\webfolder\test.php on line 6
```

1.6.3 Error Logging

- By default, PHP sends an error log to the server's logging system or a file, depending on how the error_log configuration is set in the php.ini file.
- By using the error_log() function you can send error logs to a specified file or a remote destination.
- Sending error messages to yourself by e-mail can be a good way of getting notified of specific errors.

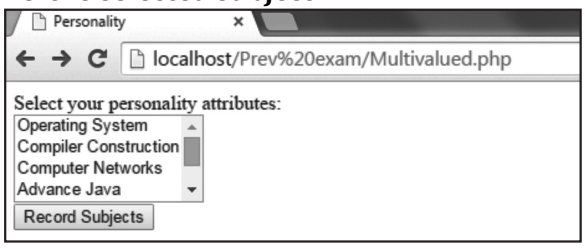
ADDITIONAL PROGRAMS

Program 1: Program to select list of subjects (use multivalued parameter) display on the next page.

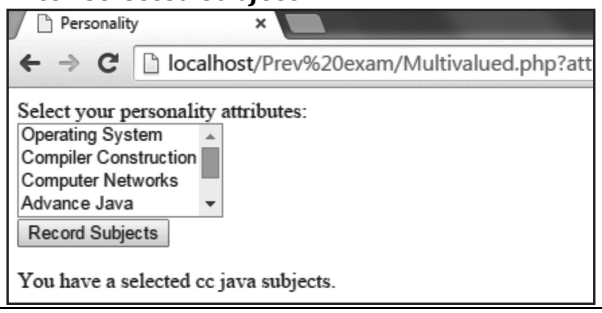
```
<html>
   <head><title>Personality</title></head>
   <body>
      <form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="GET">
         Select your personality attributes:<br />
         <select name="attributes[]" multiple>
            <option value="os">Operating System</option>
            <option value="cc">Compiler Construction</option>
            <option value="networks">Computer Networks</option>
            <option value="java">Advance Java</option>
            <option value="php">Internet Programming</option>
            <option value="graphics">Computer Graphics</option>
         </select>
         <br>
         <input type="submit" name="s" value="Record Subjects" />
      </form>
   <?php
      if (array_key_exists('s', $_GET)) {
         $description = join (" ", $_GET['attributes']);
         echo "You have a selected $description subjects.";
         }
      ?>
   </body>
</html>
```

Output:

Before selected subject:



After selected subject:



Program 2: Program to accept personal details of student (rno, name, class) on first page. On second page accept marks of six subjects (out of 100). On third page print marklist (rno, name, class, marks, total, percentage).

5b2.html

```
<html>
   <form action="5b2.php" method="POST">
      ENTER THE NAME: <input type="text" name="name">
   <br><br>
      ENTER THE CLASS: <input type="text" name="class">
   <br><br>
      ENTER THE ADDRESS: <input type="text" name="address">
   <br><br>
      <input type="submit" value="ok">
   </form>
</html>
```

5b2.php

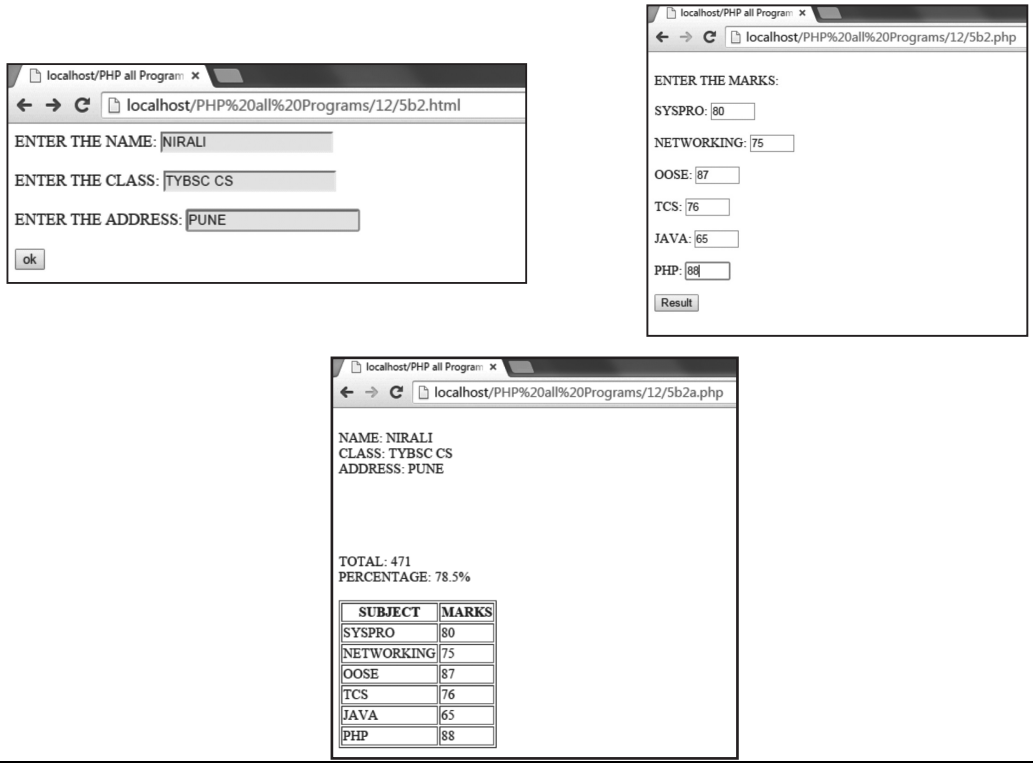
```
<?php
   setcookie('name',$_POST['name']);
   setcookie('class',$_POST['class']);
   setcookie('address',$_POST['address']);
?>
<html>
   <form action="5b2a.php" method="POST">
   <br>
      ENTER THE MARKS: <br><br>
      SYSPRO: <input type="text" name="syspro" size=3>
   <br><br>
      NETWORKING: <input type="text" name="networking" size=3>
   <br><br>
```

```
        OOSE: <input type="text" name="oose" size=3>
    <br><br>
        TCS: <input type="text" name="tcs" size=3>
    <br><br>
        JAVA: <input type="text" name="java" size=3>
    <br><br>
        PHP: <input type="text" name="php" size=3>
    <br><br>
        <input type="submit" value="Result">
    </form>
</html>
```

5b2a.php

```
<?php
    echo "<br>NAME: ".$_COOKIE["name"];
    echo "<br>CLASS: ".$_COOKIE["class"];
    echo "<br>ADDRESS: ".$_COOKIE["address"];
    echo "<br><br>";
    echo "<table border=1><tr><th>SUBJECT<th>MARKS</tr>";
    echo "<tr><td>SYSPRO<td>".$_POST["syspro"]."</tr>";
    echo "<tr><td>NETWORKING<td>".$_POST["networking"]."</tr>";
    echo "<tr><td>OOSE<td>".$_POST["oose"]."</tr>";
    echo "<tr><td>TCS<td>".$_POST["tcs"]."</tr>";
    echo "<tr><td>JAVA<td>".$_POST["java"]."</tr>";
    echo "<tr><td>PHP<td>".$_POST["php"]."</tr>";
    echo "<br>";
    echo "<br><br>";
    $total=$_POST["syspro"]+$_POST["networking"]+$_POST["oose"]+
                                $_POST["tcs"]+$_POST["java"]+$_POST["php"];
    $per=$total/6;
    echo "<br>TOTAL: ".$total;
    echo "<br>PERCENTAGE: ".$per."%";
    echo "<br><br>";
?>
```

Output:



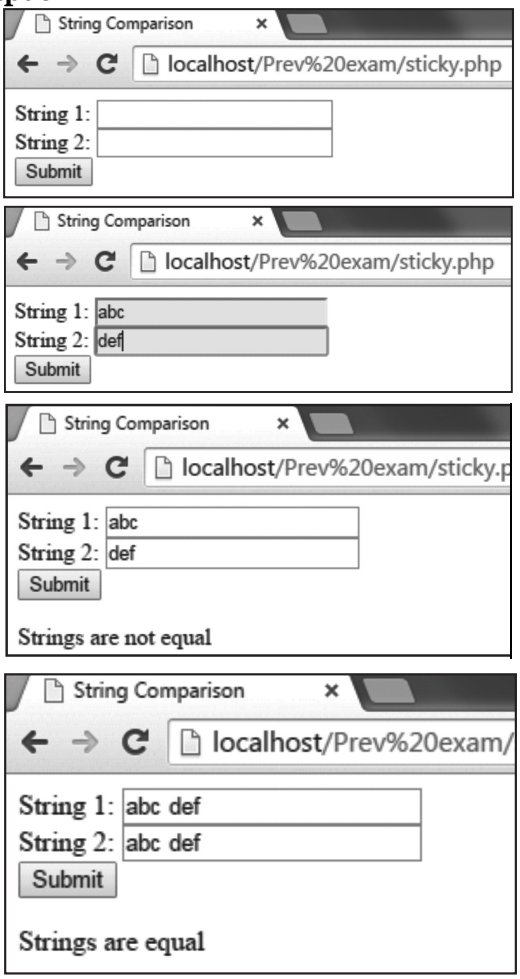
Program 3: Program to accept two strings and check if strings are equal using sticky form.

Sticky.php

```
<html>
   <head><title>String Comparison</title></head>
   <body>
      <?php
         $s1 = @$_GET['str1'];
         $s2 = @$_GET['str2'];
         ?>
         <form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="GET">
            String 1:
            <input type="text" name="str1" value="<?php echo $s1 ?>" />
            <br/>
            String 2:
            <input type="text" name="str2" value="<?php echo $s2 ?>" />
            <br/>
            <input type="submit" name="Check Equality" />
         </form>
      <?php
```

```
            if(! is_null($s1) && !is_null($s2))
            {
               if(strcmp($s1,$s2)==0)
               {
                  echo" Strings are equal";}
               else
               {
                  echo "Strings are not equal";
               }
            }
         ?>
      </body>
   </html>
```

Output:



Program 4: Program to accept Employee details (Eno, Ename, Add.) on first page. On second page accept earning (Basic, DA, HRA). On third page Print Employee Information (Eno, Ename, Add, Basic, DA, HRA, total).

5b2.html

```
<html>
   <form action="5b2.php" method="POST">
      ENTER THE Emp No: <input type="text" name="no">
   <br><br>
      ENTER THE Emp Name: <input type="text" name="name">
   <br><br>
      ENTER THE ADDRESS: <input type="text" name="address">
   <br><br>
      <input type="submit" value="ok">
   </form>
</html>
```

5b2.php

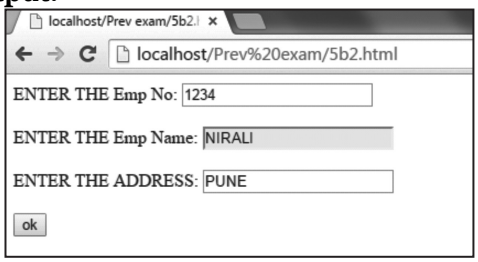
```
<?php
   setcookie('name',$_POST['no']);
   setcookie('class',$_POST['name']);
   setcookie('address',$_POST['address']);
?>
<html>
   <form action="5b2a.php" method="POST">
   <br>
      ENTER THE MARKS: <br><br>
      BASIC SALARY: <input type="text" name="basic" size=3>
   <br><br>
      HRA: <input type="text" name="hra" size=3>
   <br><br>
      DA: <input type="text" name="da" size=3>
   <br><br>
      <input type="submit" value="Display Info.">
   </form>
</html>
```

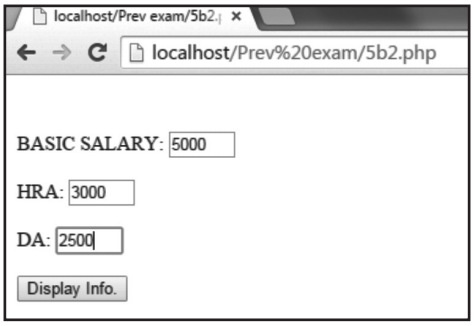
5b2a.php

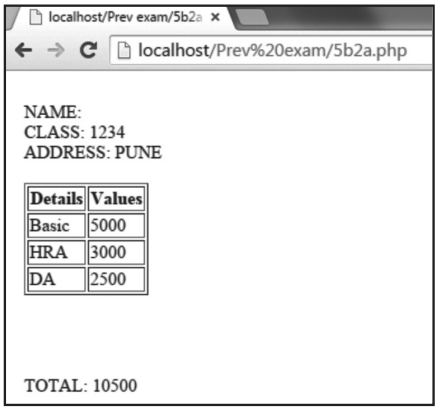
```
<?php
   echo "<br>NAME: ".$_COOKIE["no"];
   echo "<br>CLASS: ".$_COOKIE["name"];
```

```
    echo "<br>ADDRESS: ".$_COOKIE["address"];
    echo "<br><br>";
    echo "<table border=1><tr><th>Details</th><th>Values</th></tr>";
    echo "<tr><td>Basic</td><td>".$_POST["basic"]."</td></tr>";
    echo "<tr><td>HRA</td><td>".$_POST["hra"]."</td></tr>";
    echo "<tr><td>DA</td><td>".$_POST["da"]."</td></tr></table>";
    echo "<br>";
    echo "<br><br>";
       $total=$_POST["basic"]+$_POST["hra"]+$_POST["da"];
    echo "<br>TOTAL: ".$total;
?>
```

Output:







Program 5: Program to keep track of number of times the web page has been accessed.

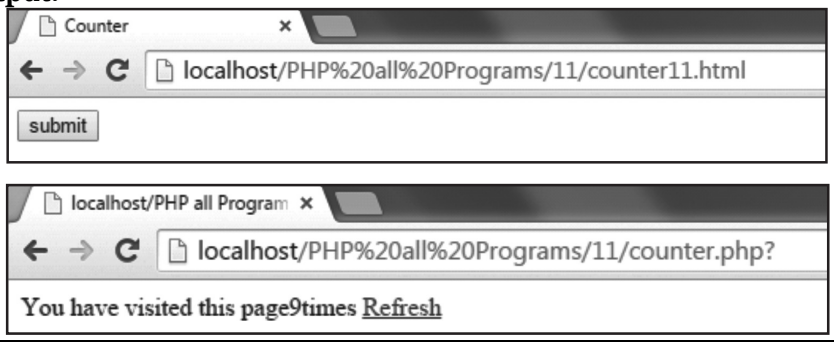
Counter11.html

```
<html>
   <head><title>Counter</title></head>
      <form method=GET action=counter.php>
         <input type=submit value=submit></input>
      </form>
</html>
```

Counter.php

```
<?php
   session_start();
   if(isset($_SESSION['cnt']))
   {
      $_SESSION['cnt']+=1;
      echo "You have visited this page".$_SESSION['cnt']."times";
   }
   else
   {
      $_SESSION['cnt']=1;
      echo "You have visited this page {$_session['cnt']} times";
   }
?>
<html>
   <body>
      <a href="counter.php">Refresh</a>
   </body>
</html>
```

Output:

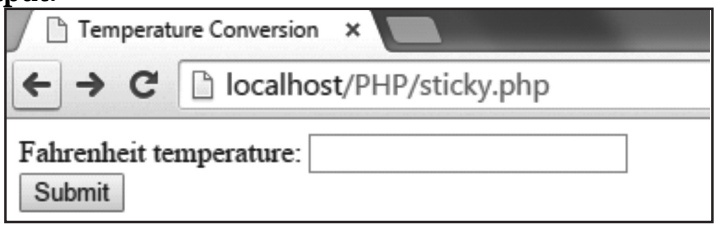


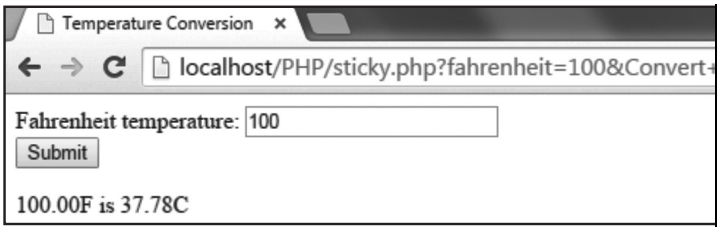
Program 6: Program to to convert temperature farhenite to celcius using sticky form.

Sticky.php

```
<html>
   <head><title>Temperature Conversion</title></head>
   <body>
      <?php
         $fahr = @$_GET['fahrenheit'];
      ?>
      <form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="GET">
         Fahrenheit temperature:
         <input type="text" name="fahrenheit" value="<?php echo $fahr?>"/>
         <br/>
         <input type="submit" name="Convert to Celsius!" />
      </form>
      <?php
         if(! is_null($fahr))
         {
            $celsius = ($fahr - 32) * 5/9;
            printf("%.2fF is %.2fC", $fahr, $celsius);
         }
      ?>
   </body>
</html>
```

Output:



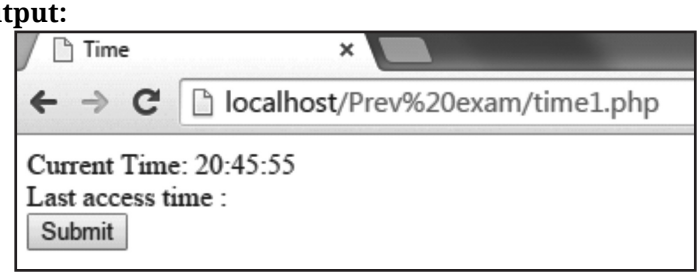


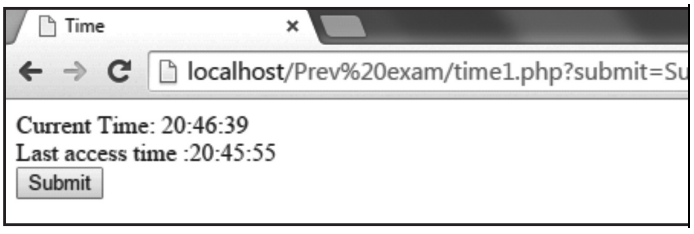
Program 7: Program that uses cookies to remember how long ago a visitor first visited the page. Display value of the page visit in minutes and seconds. (Hint: used time() to find time of cookies).

Time1.php

```
<html>
   <head><title>Time</title></head>
<body>
   <?php
      setcookie('access_time',date('H:i:s',time())); //H:i:s is Hr Min Sec
   ?>
   <form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="GET">
   <?php
      echo "Current Time: ".date('H:i:s',time());
      echo "<br>";
      echo "Last access time :";
      if(isset($_COOKIE['access_time']))
         echo "".$_COOKIE['access_time'];
         echo "<br>";
      ?>
         <input type="submit" name="submit" />
      </form>
   </body>
</html>
```

Output:





PRACTICE QUESTIONS

Q. I Multiple Choice Questions:

1. Which variables are used to keep track of information like server configuration, user request information etc.
   (a) global (b) session
   (c) tracking (d) None of the mentioned
2. Which one of the following function is used to start a session??
   (a) start_session() (b) session_start()
   (c) session_begin() (d) begin_session()
3. Which is a function that takes no argument s and return a string, which is unique key corresponding to a particular session?
   (a) session_name() (b) session_params()
   (c) session_id() (d) s_id()
4. To remove all global session variables , use
   (a) unset() (b) unset_session()
   (c) $unset() (d) Session_unset()
5. Which of the following is a not PHP global array?
   (a) $_COOKIE (b) $_GET
   (c) $_SERVICE (d) $_FILES
6. Which of the following is not a argument of cookies?
   (a) path (b) domain
   (c) secure (d) time
7. Which are used to communicate between users and the server?
   (a) sessions (b) forms
   (c) variables (d) None of the mentioned
8. Which method sends the encoded user information appended to the page request (to the URL)?
   (a) POST (b) REQEST
   (c) GET (d) SERVER
9. Which file is an initialization file and is used to configure the behaviour of PHP?
   (a) php.ini (b) php.init
   (c) php.exe (d) php.iniexe

10. In which forms the values entered by user remain displayed with the form component, if we display the form after submit.
    (a) server (b) sticky
    (c) Web (d) None of yhe mentioned
11. Which array is used in PHP to handle file uploads?
    (a) $_SERVER (b) $_GET
    (c) $_ENV (d) $_FILES
12. Which is a small piece of data or information that is sent by a Web server and stored by the browser on the client machine?
    (a) sessions (b) forms
    (c) cookies (d) All of the mentioned
13. Which techniques are used in PHP for maintaining states?
    (a) Hidden Fields (b) URL Rewriting
    (c) Cookies (d) All of the mentioned
14. Which is used to store and pass information from one page to another temporarily until user close the Website?
    (a) sessions (b) forms
    (c) cookies (d) All of the mentioned
15. To create custom error handler first use,
    (a) get_error_handler() (b) setError_handler()
    (c) set_error_handler() (d) create_error_handler()

Answers

1. (a)	2. (b)	3. (c)	4. (d)	5. (c)	6. (d)	7. (b)	8. (c)	9. (a)	10. (b)
11. (d)	12. (c)	13. (d)	14. (a)	15. (c)					

Q. II Fill in the Blanks:

1. The _____ determine whether or not to register the EGPCS variables as global variables.
2. The _____ global array contains information about any uploaded files.
3. PHP also creates a variable called _____, which holds the name of the current script, relative to the document root.
4. In _____ forms the values entered by user remain displayed with the form component.
5. _____ is created at server side and saved to client browser.
6. The Web runs on _____.
7. _____ is a way to get information of the user to the server and let the server do something in response to the user's input.

8. For client side validation _____ language is used.
9. _____ means to send the user to a new URL.
10. The _____ function to send a cookie to the browser.
11. A _____ is a series of related interactions between a single client and the web server, which takes place over a period of time.
12. By default, PHP sends an error log to the server's logging system or a file, depending on how the _____ configuration is set in the php.ini file.
13. The _____ function is used in PHP to triggering errors.
14. Selection of multiple values for one HTML control is done by using _____.
15. To keep the track of user requests we require maintaining _____.

Answers

1. register_globals	2. $_FILES	3. $PHP_SELF	4. sticky
5. Cookie	6. HTTP	7. Form	8. JavaScript
9. Redirection	10. setcookie()	11. session	12. error_log()
13. trigger_error()	14. []	15. state	

Q. III State True or False:

1. HTTP is stateless protocol because each request is executed independently, without any knowledge of the requests that were executed before it.
2. A single PHP program can be used to both generate a form and process it.
3. In PHP the form parameters are available in the $_GET and $_POST arrays.
4. Error handling is the process of catching errors raised by the program and then taking appropriate action.
5. It is not possible to make multiple selected form elements sticky.
6. PHP session is used to store and pass information from one page to another temporarily.
7. To create custom error handler first we can use the library function set_error_handler().
8. The php.ini file is a initialization file and is used to configure the behavior of PHP.
9. The $_SESSION is an associative array that contains all session variables.
10. By using the error_log() function you can send error logs to a specified file or a remote destination.
11. To make form sticky we use the submitted form value as the default value when creating the HTML field.
12. To remove all global session variables and destroy the session, use session_del() functions.

13. The elements of the $_FILES array gives information about the uploaded file.
14. Session, Cookie, Hidden form fields and URL rewriting are the techniques used to maintain the state.
15. When a browser sends a cookie back to the server, we can access that cookie through the $_ENV array.
16. When a Web browser (client) requests a web page, it sends an HTTP request to a web server then the Web server responds with a reply.
17. The $_COOKIE is a global array contains any cookie values passed as part of the request, where the keys of the array are the names of the cookies.
18. The session_start() function is used to start the session.

Answers

1. (T)	2. (T)	3. (T)	4. (T)	5. (F)	6. (T)	7. (T)	8. (T)	9. (T)	10. (T)
11. (T)	12. (F)	13. (T)	14. (T)	15. (F)	16. (T)	17. (T)	18. (T)		

Q. IV Answer the following Questions:

(A) Short Answer Questions:

1. Define variable.
2. Give the purpose of form.
3. List predefined variables in PHP.
4. What is the purpose of POST and GET methods in forms in PHP?
5. Which file is used to configure the behaviour of PHP??
6. Define sticky form.
7. Give the purpose of error_log().
8. Which array is used for file uploading?
9. Give the purpose of header().
10. Define cookies.
11. What is the use of trigger_error().
12. Define session.
13. Define error.
14. Enlist techniques for maintain the state.

(B) Long Answer Questions:

1. What are variables in PHP? Explain in detail.
2. What is form? How to create and process it? Explain wit example.
3. What is error and error handler?
4. How to get server information? Describe ib detail.
5. Describe GET and POST method with example. Also differentiate them.

6. Write short note on: Automatic quoting of parameters.
7. Explain self processing pages in detail.
8. With the help of example sticky forms.
9. What is meant by multi-valued and sticky multi-valued parameters? Explain with example.
10. How to file uploads in PHP? Example.
11. How to validate forms in PHP? Explain with example.
12. With the help of example explain how to set response headers?
13. How to maintaining state? Explain in detail.
14. What is cookie? How it works? Explain diagrammatically.
15. What is session? How to start it? Explain with example.
16. With the help of example explain how to create custom error handler.
17. Differentiate between cookie and session. Ay four points.
18. How to trigger and log an error?

UNIVERSITY QUESTIONS AND ANSWERS

April 2016

1. Which function is used to determine whether file was uploaded? **[1 M]**

Ans. Refer to Section 1.3.8.

2. State the difference between GET and POST method? **[5 M]**

Ans. Refer to Sections 1.3.1 and 1.3.2.

3. Write short note on cookies. **[4 M]**

Ans. Refer to Section 1.5.1.

4. Write PHP script to accept string and check whether it is palindrome or not by using sticky forms. **[5 M]**

Ans. Refer to Additional Programs.

April 2017

1. How to set response header in PHP? **[1 M]**

Ans. Refer to Section 1.4.

2. How to register a variable into a session? **[1 M]**

Ans. Refer to Section 1.5.2.

3. What is sticky forms? Explain with example. **[5 M]**

Ans. Refer to Section 1.3.5.

4. Explain the concept of session handling with example. **[4 M]**

Ans. Refer to Section 1.5.2.

October 2017

1. What are 'automatic globals' variables in PHP? List any four. **[1 M]**

Ans. Refer to Section 1.1.

2. List the items available in the $_FILES array. **[1 M]**

Ans. Refer to Section 1.1, Point (4).

3. How we can get the cookie values and destroy the cookies. **[1 M]**

Ans. Refer to Section 1.5.1.

4. Explain different methods used to maintain state in PHP. **[5 M]**

Ans. Refer to Section 1.5.

5. Write a PHP script by using multivalued parameter check boxes, select list of subjects and display it on next page by using sticky forms. **[5 M]**

Ans. Refer to Additional Programs.

April 2018

1. Superglobals cannot be used as variable variables inside functions or class methods. Justify true or false. **[1 M]**

Ans. Refer to Section 1.1.

2. Which information is stored by $_FILES? **[1 M]**

Ans. Refer to Sections 1.1, Point (4).

3. Discuss any five elements of $-SERVER superglobal variable. **[5 M]**

Ans. Refer to Section 1.2.

4. Write a PHP script to accept two strings. Concatenate second string to first string by using sticky forms. **[5 M]**

Ans. Refer to Additional Programs.

October 2018

1. Define cookie. **[1 M]**

Ans. Refer to Section 1.5.1.

2. Explain $_FILE array with suitable example. **[4 M]**

Ans. Refer to Section 1.3.8.

3. Write a PHP script to accept two strings from the user and check if two strings are equal or not using Sticky forms. **[5 M]**

Ans. Refer to Additional programs.

April 2019

1. Write an elements of global array $_SERVER. **[1 M]**

Ans. Refer to Section 1.1, Point (5).

2. What type of information is stored by $_FILES array? **[1 M]**

Ans. Refer to Section 1.3.8.

3. Write php script to select list of subjects i.e. O.S., TCS, N/W, PHP, JAVA, Graphics [Use multivalue parameter], display list on next page. **[5 M]**

Ans. Refer to Section 1.3.6.

4. What is sticky form? Explain with example. **[4 M]**

Ans. Refer to Section 1.3.5.

■■■

CHAPTER 2

XML

Objectives...

- To understand XML
- To learn XML Document Structure
- To study PHP and XML
- To understand SimpleXML

2.0 INTRODUCTION

- XML (eXtensible Markup Language) is a text-based markup language derived from Standard Generalized Markup Language (SGML).
- The design goals of XML emphasize simplicity, generality, and usability across the Internet. XML is a markup language that defines set of rules for encoding documents in a format that is both human-readable and machine-readable.

2.1 WHAT IS XML? [April 16, 17, 18, 19, Oct. 17, 18]

- The XML (eXtensible Markup Language) lets us to create text documents that can hold data in a structured way. Actually XML is a means of storing structured data.
- Although XML is different from a database in many ways, both XML and databases offer ways to format and store structured data. XML is not a language but a specification for creating your own markup languages.
- It is a subset of Standard Generalized Markup Language (SGML, the parent of HTML). XML can exchange data easily from one application to another, even if both the applications are different.
- XML is much like HTML. The difference is:
  1. HTML tags are predefined, XML tags are not predefined.
  2. HTML is used to display data but XML is used to store and transport data.
- The XML - based language can be placed online and any application can read and write it. So, two applications that know nothing about each other can still exchange data.

- For these reasons XML is rapidly becoming the data exchange standard, and many useful technologies have been created using XML, such as:
  1. Web Services, including languages such as SOAP for exchanging information in XML format over HTTP.
  2. XML - RPC, and the Web Services Description Language (WSDL), used for describing Web Services.
  3. Application file formats, such as OpenOffice's OpenDocument Format (ODF) and Microsoft's Office Open XML (OOXML) that are used to store word processing documents, spreadsheets, and so on.
  4. RSS and Atom news feeds that allow Web applications to publish news stories in a universal format that can be read by many types of software, from news readers and email clients through other Web site applications.
  5. WAP and WML, markup languages for handheld devices.
  6. SMIL, describing multimedia for the web.
- Like HTML, an XML document contains elements and attributes in the form of tags.

Characteristics of XML:

1. **XML is extensible:** XML allows you to create your own self-descriptive tags, or language, that suits your application.
2. **XML is a public standard:** XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.
3. **XML carries the data, does not present it:** XML allows you to store the data irrespective of how it will be presented.

- Though XML documents are human - readable, many applications are designed to parse XML documents automatically and work efficiently with their content. PHP has many XML - related functions that can easily be used to work with XML documents.
- We can make your own XML document as easily as this:

```
<?xml version="1.0" ?>
   <stockList>
      <item type="fruit">
         <name>apple</name>
         <unitPrice>0.99</unitPrice>
         <quantity>412</quantity>
      </item>
         <item type="vegetable">
            <name>beetroot</name>
            <unitPrice>1.39</unitPrice>
            <quantity>67</quantity>
         </item>
   </stockList>
```

- The first line of this document is called the **XML declaration**; it indicates that this document is an XML document, and specifies the version of XML that is used to create the document.
- The second line defines the **root element** of the document (named stockList). There can be only one root element for an XML document.
- The third line defines a **child element named item**, it contains an attribute named type that is set to the value fruit. Apart from elements and attributes, XML also contains plain text such as apple, 0.99 etc.
- From reading this XML document, you can tell that:
  - It stores a list of stock items.
  - There are 412 apples available, and an apple is a fruit and costs $0.99.
  - There are 67 beetroots available, and a beetroot is a vegetable and costs $1.39.
- While creating elements and attributes, their meaning and structure is also specified. This is so that, when we exchange data with another application, both parties to the transaction know exactly what the element and attribute names mean.
- To do this, we use either a Document Type Definition (DTD) or an XML Schema Definition (XSD).
- Frequently when you create XML documents, you will either use an existing publicly available DTD (or XSD) or the DTD written by yourself.
- Once, we write a DTD, you can publish it on the Web. That means anyone who needs to read or write an XML document compatible with the system has the capability to access the published DTD to make sure the document is valid.

2.2 XML DOCUMENT STRUCTURE [April 17]

- XML document is a well - formed and valid document.
- A well - formed XML document follows the basic XML syntax rules, and a valid document also follows the rules which are defined in a DTD or an XML schema.
- In other words,
  1. **All XML documents must be well-formed:** A well - formed XML document uses correct XML syntax. Means any elements, attributes, or other constructs are as per the XML specification.
  2. **An XML document can also be valid:** A well - formed document does not need to be valid, but a valid document must be well - formed. An XML document is valid if its elements, attributes, and other contents follow the rules in the DTD or an XML schema.
- By using valid XML documents, applications that know nothing about each other can still communicate effectively - they just have to exchange XML documents and understand the meaning of the DTD or XML schema against which those documents are validated. This is one of the main features that make XML so powerful.

2.2.1 Major Parts of an XML Document

- A well - formed XML document may contain the following:
- First line of an XML document is an XML version line; possibly include a character encoding declaration. Character encoding declaration is optional. If no character encoding is specified, UTF - 8 is assumed. For example:

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

- An optional DTD or an XML schema, or a reference to one of these if they are stored externally. This must appear before the document's root element. For example, here's a reference to an external DTD:

```
<!DOCTYPE stockList SYSTEM "http://www.example.com/dtds/stockList.dtd">
```

- All XML documents must contain one and only one - root element. This element usually contains one or more child elements, each of which may optionally have one or more attributes.
- An element can contain other child elements or data between its beginning and ending tag, or it may be empty.
- XML documents may contain additional components such as processing instructions (PIs), CDATA sections, comments; entity references; text; and entities.

2.2.2 XML Syntax Rules [April 16, 17, 18]

- A well - formed XML document must follow all the syntax rules of the XML specification, the most common of which are:
  1. There is only one parent element containing all the rest of the elements.
  2. XML elements are declared to be either non - empty, in which case they are designed to contain data; or empty, in which case they cannot contain data. For example, in HTML, the ... element is non - empty because it can contain text, whereas the
 (line - break) element is empty because it cannot contain anything.
  3. XML elements must have a closing tag.
     For example:

     ```
     <p>This is a paragraph      - In HTML
     <p>This is a paragraph</p> - In XML
     ```

  4. XML tags are case sensitive. **[April 16]**
     For example:

     ```
     <Message>This is incorrect</message>
     <message>This is correct</message>
     ```

  5. XML elements can have attributes in name/value pairs like HTML. In XML the attribute values must always be quoted.

For example:

```
<books>
    <book id=”1”>PHP 6</book>
<books>
```

- In an element attribute name can not be repeated.
- XML elements must be properly nested. For example:

```
<!-- Incorrect nesting -->
<parent> <child> </parent> </child>
<!-- Correct nesting -->
<parent> <child> </child> </parent>
```

- Names must start with a letter, an underscore, or a colon, but in practice, you should never use colons unless you are dealing with XML namespaces.
- Names are case-sensitive. Letters, numbers, the hyphen, the underscore, and the period can be used after the first character.
- Comments are delimited in the same way as HTML comments i.e.,

```
<!--   and   -->
```

2.2.3 Predefined Character Entities

- Some character have a special meaning in XML If you use ‘<’ character inside an XML element, it will give an error because the character ‘<’ is used as a start of a new element.

  For example: The following code will generate an XML error:

```
<message>If salary < 1000 then</message>
```

- To remove this error, replace ‘<’ with their entity reference.

  For example: The correct code is as follows,

```
<message>If salary &lt; 1000 then</message>
```

- Hence, the entity reference for these predefined characters are as follows:

>	>
<	<
‘	'
“	"
&	&

2.2.4 CDATA Sections

- The term CDATA means, Character Data.
- CDATA are defined as, blocks of text that are not parsed by the parser, but are otherwise recognized as markup.

Syntax:

```
<![CDATA[
   characters with markup
]]>
```

- The above syntax is composed of three sections:
  - **CDATA Start section:** CDATA begins with the delimiter `<![CDATA[`
  - **CDATA End section:** CDATA section ends with `]]>` delimiter.
  - **CData section:** Characters between these two delimiters are interpreted as characters, and not as markup. This section may contain markup characters (<, >, and &), but they are ignored by the XML processor.
- The following markup code shows example of CDATA. Here, each character written inside the CDATA section is ignored by the parser.

```
<script>
<![CDATA[
   <message> Welcome to Nirali Prakashan </message>
]] >
</script >
```

- In the above syntax, everything between <message> and </message> is treated as character data and not as markup.

2.2.5 XML Processing

- Processing Instructions (PIs) allow documents to contain instructions for applications.
- Processing instructions (PIs) can be used to pass information to applications.

  Syntax: `<?target instructions?>`

  where, target identifies the application to which the instruction is directed. Instruction is a character that describes the information for the application to process.
- The following line shows a style sheet "niraliprakashan.css" is linked with XML document. So browser will display the XML document according to the CSS.

```
<?xml-stylesheet href="niraliprakashan.css" type="text/css"?>
```

- Here, the target is xml-stylesheet. href="niraliprakashan.css" and type="text/css" are data or instructions that the target application will use at the time of processing the given XML document.

Example:

```
<?xml version=”1.0” ?>
<?xml-stylesheet href="course.css" type="text/css"?>
<Course>
   <Computer Science>
```

```
            <Student name> Bianca</Student name>
            <Class name>TYBsc </Class name>
            <percentage>76.4</percentage>
        </Computer Science>
    </Course>
```

2.2.6 XML Tree Structure

- An XML document can also be described as a tree structure.
- For example consider the following XML document:

```
<?xml version="1.0"?>
<Company>
  <Employee>
      <FirstName>Tanmay</FirstName>
      <LastName>Patil</LastName>
      <ContactNo>1234567890</ContactNo>
      <Email>tanmaypatil@xyz.com</Email>
      <Address>
           <City>Bangalore</City>
           <State>Karnataka</State>
           <Zip>560212</Zip>
      </Address>
  </Employee>
</Company>
```

- Following tree structure represents the above XML document:

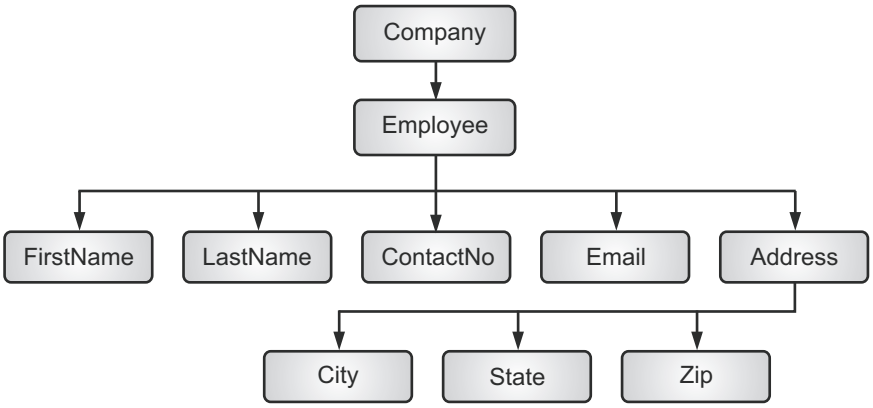


Fig. 2.1: XML Tree structure

- In the Fig. 2.1, there is a root element named as <company>. Inside that, there is one more element <Employee>.

- Inside the employee element, there are five branches named <FirstName>, <LastName>, <ContactNo>, <Email> and <Address>.
- Inside the <Address> element, there are three sub-branches, named <City> <State> and <Zip>.

2.3 PHP AND XML

- PHP has functions and classes to make it easier to work with XML documents. Using PHP you can read XML document, you can also add more element to the document, and also modify and remove elements from the document.
- In the coming section you are going to do the following things with PHP and XML:
  - Reading, or parsing, XML documents using the XML Parser extension.
  - Using the DOM extension to manipulate XML documents via the Document Object Model.
  - Reading, writing, and manipulating XML documents using SimpleXML extension.
- Before you start parsing and using PHP predefined functions, you can generate XML manually i.e. without using PHP built-in function.
- **Example:** A PHP script to generate an XML document.

```
<?php
   header('Content-Type: text/xml');
      print '<?xml version="1.0"?>' . "\n";
      print "<Course>\n";
   $ComputerScience = array(array('StudentName' => 'Bianca',
   'ClassName' => 'TYBSc', 'percentage' => '76.4'), array('StudentName' =>
   'Bhupesh', 'ClassName' => 'TYBSc', 'percentage' => '72.0'));
   foreach ($ComputerScience as $cs)
   {
         print " <ComputerScience>\n";
      foreach($cs as $tag => $data)
      {
         print " <$tag>" . htmlspecialchars($data) . "</$tag>\n";
      }
         print " </ComputerScience>\n";
   }
      print "</Course>\n";
?>
```

Output:

```
<?xml version="1.0"?>
<Course>
   <ComputerScience>
      <StudentName>Bianca</StudentName>
      <ClassName>TYBSc</ClassName>
      <percentage>76.4</percentage>
   </ComputerScience>
   <ComputerScience>
      <StudentName>Bhupesh</StudentName>
      <ClassName>TYBSc</ClassName>
      <percentage>72.0</percentage>
   </ComputerScience>
</Course>
```

- In above program a 2D array is created to store the child elements of "Course". Hence, nested for each loop is used to display the elements.

2.4 XML PARSER [Oct. 17, April 19]

- XML parser is used to create, read and manipulate an XML document.
- In PHP there are two major types of XML parsers: **[April 19]**
  1. **Tree-Based Parsers:** Tree-based parser transforms the XML document into a Tree structure, and then you can access the tree elements individually. Example of tree-based parsers: SimpleXML and DOM.
  2. **Event-Based Parsers:** View an XML document as a series of events. When a specific event occurs, it calls a function to handle it. Event-based parsers do not hold the entire document in Memory; instead, they read in one node at a time and allow us to interact with in real time. Once we move onto the next node, the old one is thrown away. Event based parsers focus on the content of the XML document, not their structure. So it parses faster and consumes less memory. Example of event-based parsers namely, XMLReader and XML Expat Parser.

2.4.1 Creating a New Parser

- This section shows how to create an Event based parser to read XML documents. For these perform the following steps:

  Step 1 : Create a new parser resource by calling the xml_parser_create() function.

  Step 2 : Create two event handler functions to handle the start and end of an XML element, then register these functions with the parser using the xml_set_element_handler() function.

Step 3 : Create another event handler function to handle any character (text) data that may be found inside an element, and register this function with the parser using xml_set_character_data_handler() .

Step 4 : Parse the XML document by calling the xml_parse() function, passing in the parser and the XML string to parse.

Step 5 : Finally, destroy the parser resource, if it is no longer needed, by calling xml_parser_free().

- These functions and few more functions are described as follows:
  - `xml_parser_create()`: Create an XML parser. Generate a new parser resource that can be used with other functions.

    ```
    $parser = xml_parser_create();
    ```
  - `xml_parser_free()`: To free up the resource when done.
  - `xml_parse_into_struct()`: Parse XML data into an array structure.
  - `xml_get_error_code()`: Get XML parser error code.
  - `xml_error_string()`: Get the textual description of the error based on error code.
  - `xml_set_element_handler()`: Set up start and end element handler.
  - `xml_set_character_data_handler()`: Set up character data handler.
  - `xml_get_current_line_number()`: Gets current line number for an XML parser.
  - `xml_parse()`: Start parsing XML document.
- **Example:** First create an XML document.

```
<?xml version="1.0" encoding="UTF-8"?>
   <stockList>
      <item type="fruit">
         <name>apple</name>
         <unitPrice>0.99</unitPrice>
         <quantity>412</quantity>
      </item>
      <item type="vegetable">
         <name>beetroot</name>
         <unitPrice>1.39</unitPrice>
         <quantity>67</quantity>
      </item>
   </stockList>
```

- Save this XML document as “stock_list.xml”. Now you read this document using the following parser script.

```
<pre>
<?php
   //Start element handler
   function startElementHandler($parser, $element, $attributes)
   {
      echo "Start of element: \"$element\"";
      if($attributes)
         echo ", attributes: ";
         foreach($attributes as $name => $value)
            echo "$name=\"$value\" ";
            echo "\n";
   }
   //End element handler
   function endElementHandler($parser, $element)
   {
      echo "End of element: \"$element\"\n";
   }
   function characterDataHandler($parser, $data)
   {
      if(trim($data))
         echo " Character data: \"" . htmlspecialchars($data) . "\"\n";
   }
   //Error handler:
   function parseError( $parser )
   {
      $error = xml_error_string( xml_get_error_code( $parser ) );
      $errorLine = xml_get_current_line_number( $parser );
      $errorColumn = xml_get_current_column_number( $parser );
      return "<b>Error: $error at line $errorLine column
                                                        $errorColumn</b>";
   }
   // Create the parser and set options
   $parser = xml_parser_create();
   xml_parser_set_option( $parser, XML_OPTION_CASE_FOLDING, false );
```

```
    // Register the event handlers with the parser
    xml_set_element_handler($parser, "startElementHandler",
                                                "endElementHandler");
    xml_set_character_data_handler( $parser, "characterDataHandler");
    // Read and parse the XML document
       $xml = file_get_contents( "stock_list.xml" );
       xml_parse( $parser, $xml ) or die( parseError( $parser ) );
       xml_parser_free( $parser );
?>
</pre>
```

Output:

```
Start of element: "stockList"
Start of element: "item", attributes: type="fruit"
Start of element: "name"
Character data: "apple"
End of element: "name"
Start of element: "unitPrice"
Character data: "0.99"
End of element: "unitPrice"
Start of element: "quantity"
Character data: "412"
End of element: "quantity"
End of element: "item"
Start of element: "item", attributes: type="vegetable"
Start of element: "name"
Character data: "beetroot"
End of element: "name"
Start of element: "unitPrice"
Character data: "1.39"
End of element: "unitPrice"
Start of element: "quantity"
Character data: "67"
End of element: "quantity"
End of element: "item"
End of element: "stockList"
```

- This program have three event handler function startElementHandler(), endElementHandler(), and characterDataHandler(). The function will be called when start of element comes, end of element comes and character data comes respectively and display data accordingly. Finally, the script reads the file to parse into a variable, then calls xml_parse() to parse the variable's contents.
- **Example:** In the following program you read the XML document "stock_list.xml" and parse the XML data into an array.

```
<body>
   <h3>Reading XML document</h3>
   <pre>
   <?php
      $xml_file = "stock_list.xml";
      $xml_parser = xml_parser_create();
      $fp = fopen($xml_file, "r") or die("Could not open");
      $xml_data = fread($fp, 4096);
      xml_parse_into_struct($xml_parser, $xml_data, $values);
      xml_parser_free($xml_parser);
      print_r($values);
   ?>
   </pre>
</body>
```

- After we run the program it displays all the elements and their attributes into array structure.

2.5 THE DOCUMENT OBJECT MODEL [Oct. 17]

- XML Parser can only read XML documents; it can not alter documents or create new documents.
- An alternative approach is to use the Document Object Model (DOM) extension.
- DOM is a way of expressing the various nodes (elements, attributes, and so on) of an XML document as a tree of objects. Then the tree can be traversed and various nodes can be accessed.
- By using various DOM class methods you can also change any of these nodes, and even create a new DOM document from scratch.
- The DOM class description as follows:
  1. `DOMNode`: Represents a single node in the DOM tree. Most DOM classes derive from the DOMNode class.
  2. `DOMDocument`: Stores an entire XML document in the form of a DOM tree. It derives from the DOMNode class, and is effectively the root of the tree.

3. DOMElement: Represents an element node.
4. DOMAttr: Represents an element's attribute.
5. DOMText: Represents a plain - text node.
6. DOMCharacterData: Represents a CDATA (character data) node.

- To start working with a DOM document, you first create a DOMDocument object:

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

- We can then use this object to read in or write out an XML document; examine and change the various nodes in the document; and add or delete nodes from the document's tree.
- **Example:** The following program read the same XML Document "stock_list.xml" using the DOM method.

```
<html>
   <body>
      <h3>Reading an XML File with the DOM Extension</h3>
   <pre>
   <?php
      // Read the XML document into a DOMDocument object
      $doc = new DOMDocument();
      $doc->load("stock_list.xml");
      // Traverse the document
      traverseDocument( $doc );
      //Traverses each node of the DOM document recursively
      function traverseDocument($node)
      {
         switch($node->nodeType)
         {
            case XML_ELEMENT_NODE:
               echo "Found element: \"$node->tagName\"";
            if($node->hasAttributes())
            {
               echo " with attributes: ";
               foreach($node->attributes as $attribute)
               {
                  echo "$attribute->name=\"$attribute->value\" ";
               }
            }
            echo "\n";
         break;
```

```
            case XML_TEXT_NODE:
               if(trim($node->wholeText))
               {
                  echo "Found text node: \"$node->wholeText\"\n";
               }
            break;
            case XML_CDATA_SECTION_NODE:
               if(trim($node->data))
               {
                  echo "Found character data node:
                              \"" . htmlspecialchars($node->data) . "\"\n";
               }
            break;
         }
         if($node->hasChildNodes())
         {
            foreach($node->childNodes as $child)
            {
            traverseDocument($child);
            }
            }
         }
      ?>
      </pre>
    </body>
  </html>
```

Output:

Reading an XML File with the DOM Extension

```
Found element: "stockList"
Found element: "item" with attributes: type="fruit"
Found element: "name"
Found text node: "apple"
Found element: "unitPrice"
Found text node: "0.99"
Found element: "quantity"
Found text node: "412"
```

```
Found element: "item" with attributes: type="vegetable"
Found element: "name"
Found text node: "beetroot"
Found element: "unitPrice"
Found text node: "1.39"
Found element: "quantity"
Found text node: "67"
```

2.5.1 Creating an XML Document using the DOM

- To create a node, you call various methods of the DOMDocument class. Some useful methods are:
  1. `createElement(name [, value ])`: Creates an element node called name and optionally appends a text node to it containing value.
  2. `createTextNode(content)`: Creates a text node that contains content.
  3. `createCDATASection(data)`: Creates a character data node that contains data.
  4. `createComment(data)`: Creates a comment node that contains data.
  5. `appendChild()`: Appends a child element to its root element.
  6. `setAttribute()`: Add attributes to element nodes.
  7. `createAttribute()`: Create new attribute.
  8. `saveXML()`: Converts the DOMDocument object into string.
- **Example:** Following program create an XML document using DOMDocument.

```
<html>
   <body>
      <h3> Create an XML document with DOMDocument</h3>
   <pre>
   <?php
      $xml = new DOMDOcument("1.0", "UTF-8");
      echo htmlspecialchars($xml->saveXML());
   ?>
   </pre>
   </body>
</html>
```

Output:

Create an XML document with DOMDocument

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

- The program create an XML document with only version line. First statement creates a DOMDocument object $xml which refers the XML document containing version line. To display the document to the browser you need to convert $xml to string, so that it can be displayed using echo.
- The next example shows how to add elements to the XML document.
- Following is a script to create XML file named “Course.xml”.

```
<Course>
   <Computer Science>
      <Student name>.......</Student name>
      <Class name>......</Class name>
      <percentage>.....</percentage>
      </Computer Science>
</Course>
```

- Store the details of a student who are in TYBSc. The solution is given below:

```
<?php
   // Create a DOMDocument object and set nice formatting
   $doc = new DOMDocument("1.0", "UTF-8");
   $doc->formatOutput = true;
   // Create the root "Course" element
   $course= $doc->createElement( "Course" );
   $doc->appendChild( $course);
   // Create the first element (Computer Science)
   $cs = $doc->createElement( "Computer_Science" );
   $course->appendChild( $cs );
   // Create the Student’s "name" child element
   $name = $doc->createElement( "Student_Name", "Alok" );
   $cs->appendChild( $name );
   // Create the Student’s "Class Name" child element
   $class_name = $doc->createElement( "Class_Name", "TYBSc" );
   $cs->appendChild( $class_name );
   // Create the Student’s "Percentage" child element
   $percentage = $doc->createElement( "Percentage", "76.4" );
   $cs->appendChild( $percentage );
   // Create a file Course.xml and the XML document will be
                                                    stored in that file
   $doc->save("Course.xml");
   echo "<h4>Course.xml created</h4>";
?>
```

Output:

```
Course.xml created
```

- After adding elements call save() function which is used to save the created XML document into the file specified by the parameter.

2.5.2 Adding Elements to an Existing Document

- The following example reads the "stock_list.xml" file as a DOM document, adds a new item element to the stockList element, and then outputs the modified XML.

```
<html>
   <body>
      <h3>Adding an Element to an XML File with the DOM Extension</h3>
   <pre>
   <?php
      // Load the XML file
      $doc = new DOMDocument();
      $doc-> preserveWhiteSpace = false;
      $doc-> load("stock_list.xml");
      $doc-> formatOutput = true;
      // Get the stockList root element
      $stockListElements = $doc-> getElementsByTagName("stockList");
      $stockList = $stockListElements-> item(0);
      // Create a new "item" element and add it to the stockList
      $item = $doc-> createElement("item");
      $item-> setAttribute("type", "vegetable");
      $stockList-> appendChild($item);
      // Create the item's "name" child element
      $name = $doc-> createElement("name", "carrot");
      $item-> appendChild($name);
      // Create the item's "unitPrice" child element
      $unitPrice = $doc-> createElement("unitPrice", "0.79");
      $item-> appendChild($unitPrice);
   // Create the item's "quantity" child element
   $quantity = $doc-> createElement("quantity", "31");
   $item-> appendChild($quantity);
   // Create the item's "description" child element
   $description = $doc-> createElement("description");
   $item-> appendChild($description);
   $cdata = $doc-> createCDATASection("Carrots are crunchy");
   $description-> appendChild($cdata);
   // Output the XML document, encoding markup characters as needed
   echo htmlspecialchars($doc-> saveXML());
   ?>
   </pre>
   </body>
</html>
```

Output:

Adding an Element to an XML File with the DOM Extension

```
<?xml version="1.0" encoding="UTF-8"?>
   <stockList>
      <item type="fruit">
         <name>apple</name>
         <unitPrice>0.99</unitPrice>
         <quantity>412</quantity>
      </item>
      <item type="vegetable">
         <name>beetroot</name>
         <unitPrice>1.39</unitPrice>
         <quantity>67</quantity>
      </item>
      <item type="vegetable">
         <name>carrot</name>
         <unitPrice>0.79</unitPrice>
         <quantity>31</quantity>
         <description><![CDATA[Carrots are crunchy]]></description>
      </item>
   </stockList>
```

- The saveXML() method is used converts the DOMDocument object into string, so that that can be displayed using echo.

2.6 SimpleXML

- Although the DOM extension is a powerful way to work with XML documents, but it contains large number of classes, methods and properties so, working with DOM extension can be tedious and time - consuming to code.
- Using SimpleXML is simpler than DOM. SimpleXML extension offers a more straightforward way to manipulate elements within an XML document.
- DOM extension provides more than 15 classes, but SimpleXML has only one class SimpleXMLElement.
- The DOM extension works on the node level, SimpleXML works at the element level, making element manipulation a much more straightforward process.
- Here, is a list of common SimpleXMLElement methods that you can use to manipulate XML documents:

1. `addAttribute(name, value)`: Adds an attribute named name, with the value of value, to the element.

2. `addChild(name [, value])`: Adds a child element called name to the element. The child element can be empty, or it can contain the text value. It returns the child element as a new SimpleXMLElement object.
3. `asXML([filename])`: Generates an XML document from the SimpleXMLElement object. If filename is supplied, it writes the XML to the file; otherwise it returns the XML as a string.
4. `attributes()`: Returns an associative array of all the attributes in the element, as name= > value pairs.
5. `children()`: Returns an array of all the element's children, as SimpleXMLElement objects.
6. `getName()`: Returns the name of the element as a string.
7. `xpath(path)`: Finds child elements that match the given XPath (XML Path Language) path string.

- In addition, SimpleXML gives three functions that we can use to import XML data into a SimpleXMLElement object:
  1. `simplexml_import_dom(node)`: Converts the supplied DOM node, into a SimpleXMLElement object.
  2. `simplexml_load_file(filename)`: Loads the XML file with name filename as a SimpleXMLElement object.
  3. `simplexml_load_string(string)`: Loads the supplied XML string as a SimpleXMLElement object.
- With SimpleXML, all the elements in an XML document are represented as a tree of SimpleXMLElement objects. Any given element's children are available as properties of the element's SimpleXMLElement object.
- For example, if $parent is a SimpleXMLElement object representing an element that has a child element called child, we can access that child element's text value directly with:

```
$value = $parent- > child;
```

- To do the same thing with the DOM classes would require several lines of code.

2.6.1 Reading an XML Document

- The following example shows how you can easily read and display the contents of an XML document using SimpleXML:

```
<html>
   <body>
      <h3> Reading an XML File with the SimpleXML Extension </h3>
   <pre>
      <?php
```

```
        // Read the XML document into a SimpleXMLElement object
        $stockList = simplexml_load_file("stock_list.xml");
        // Display the object
        echo htmlspecialchars($stockList->asXML());
    ?>
    </pre>
    </body>
</html>
```

Output:

Reading an XML File with the SimpleXML Extension

```
<?xml version="1.0" encoding="UTF-8"?>
    <stockList>
      <item type="fruit">
          <name>apple</name>
          <unitPrice>0.99</unitPrice>
          <quantity>412</quantity>
      </item>
      <item type="vegetable">
          <name>beetroot</name>
          <unitPrice>1.39</unitPrice>
          <quantity>67</quantity>
      </item>
</stockList>
```

- In above program first you call simplexml_load_file() function creates a reference to XML document contained by stock_list.xml file. It actually loads the root element stockList from the file stock_list.xml.
- The XML document can be displayed to the browser using echo after converting object $stockList into string. A htmlspecialchars() function is used so that the XML predefined characters can not be interpreted by the browser, and the browser can display the XML document.

2.6.2 Creating an XML Document with SimpleXML

- The following example shows how to create an XML document using SimpleXML extension.

- A script to create “cricket.xml” file with multiple elements as given below:

```
<CricketTeam>
   <Country name = “India”>
      <PlayerName >-------- <Player Name >
      <Wickets>--------- </Wickets>
      <Runs>--------</Runs>
   </Country>
</CricketTeam>
```

- Also add country = “England” and its elements.

```
<?php
   // Create the root "CricketTeam" element
   $cricketTeam = new SimpleXMLElement(" <CricketTeam/> ");
   $country = $cricketTeam-> addChild("Country");
   $country -> addAttribute("name", "India");
   $country -> addChild("PlayerName", "M Dhoni");
   $country -> addChild("Wickets", "36");
   $country -> addChild("Runs", "10000");
   $country = $cricketTeam-> addChild("Country");
   $country -> addAttribute("name", "England");
   $country -> addChild("PlayerName", "Alastair Cook");
   $country -> addChild("Wickets", "2");
   $country -> addChild("Runs", "9500");
// Save the created XML document into cricket.xml file
   $cricketTeam -> asXML("cricket.xml");
   echo "<h4>cricket.xml created</h4>";
?>
```

- Open the “criclet.xml” file and see the output.

2.7 CONVERTING BETWEEN SIMPLEXML AND DOM OBJECTS

- It is easy to convert between a SimpleXMLElement object and a DOMElement object, meaning we can work with SimpleXML if we prefer, then switch to the DOM when we need to do something that SimpleXML can not do.
- The two key functions here are:
  1. `simplexml_import_dom(node)`: Converts the supplied DOM node, node, into a SimpleXMLElement object.
  2. `dom_import_simplexml(element)`: Converts the supplied SimpleXMLElement object, element, into a DOM node.

- **Example:** The program shows the DOM object is converted into SImpleXML object.

```
<?php
   $xmlstr = <<< XML
   <php_programs>
      <program name = "Cart">
         <price> 100 </price>
      </program>
      <program name = "Survey">
         <price> 500 </price>
      </program>
   </php_programs>
XML;
   $dom = new DOMDocument;
   $dom->loadXML($xmlstr);
   $s_dom = simplexml_import_dom($dom);
   echo "The price of the first program is ".$s_dom->program[0]->price;
?>
```

Output:

```
The price of the first program is 100
```

- In above program one string variable $xmlstr is created which stores the XML document. Then DOM object is created and the XML is loaded into it.
- Now simplexml_import_dom() function converts the DOM object into the SimpleXML object $s_dom. Finally, using SimpleXML method the price of the first program “Cart” is displayed.

2.8 CHANGING A VALUE WITH SIMPLEXML

- Not only you can read the part of an XML document into a SimpleXML object, you can change data in the in-memory document.
- All you need to do is properly address the data you want to change and then set it equal to the value you want it to have.

Example:

```
<?php
   $xmlstr = <<< XML
      <php_programs>
         <program name = "Cart">
            <price> 100 </price>
```

```
            </program>
            <program name = "Survey">
               <price> 500 </price>
            </program>
         </php_programs>
      XML;
      $first_xml_string = Simplexml_load_string($xmlstr);
      $first_xml_string→program[0]→price = '250';
      echo "<pre>";
      var_dump($first_xml_string);
      echo "</pre>";
   ?>
```

Output:

```
SimpleXMLElement Object
(
    [program] => Array
        (
            [0] => SimpleXMLElement Object
                (
                    [@attributes] => Array
                        (
                            [name] => Cart
                        )
                    [price] => 250
                )
            [1] => SimpleXMLElement Object
                (
                    [@attributes] => Array
                        (
                            [name] => Survey
                        )
                    [price] =>  500
                )
        )
)
```

ADDITIONAL PROGRAMS

Program 1: Program to read emp. XML file (contains emp-no, emp-name, salary, designation) and print employee details in tabular format, (use SimpleXML).

emp.xml

```
<?xml version="1.0" encoding="utf-8"?>
<employee>
   <emp>
      <empno>111</empno>
      <empname>John</empname>
      <salary>10000</salary>
      <designation>Accountant</designation>
   </emp>
   <emp>
      <empno>120</empno>
      <empname>Dane</empname>
      <salary>15000</salary>
      <designation>Asst. Manager</designation>
   </emp>
   <emp>
      <empno>122</empno>
      <empname>Wiely</empname>
      <salary>17000</salary>
      <designation>Manager</designation>
   </emp>
   <emp>
      <empno>100</empno>
      <empname>Robert</empname>
      <salary>12000</salary>
      <designation>Sr. Accountant</designation>
   </emp>
</employee>
```

XML.html

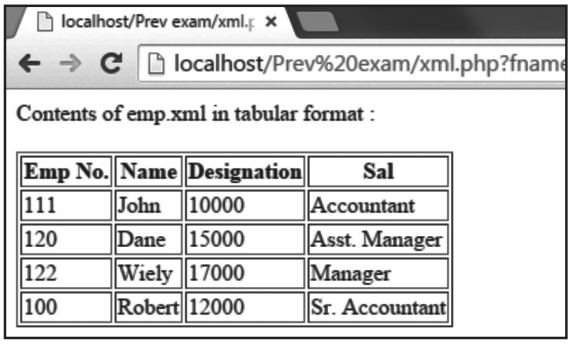
```
<html>
   <head><title>XML</title></head>
   <body>
      <form action="xml.php" method="get">
```

```
            Select XML File:
            <input type="file" name="fname" />
            <br><br>
            <input type="submit" name="submit" value="Read and Display">
         </form>
      </body>
   </html>
```

Xml.php

```
   <?php
      $fname=$_GET['fname'];
      $xml=simplexml_load_file($fname) or die("Error: Cannot create object");
      echo "Contents of $fname in tabular format : <br><br>";
      echo"<table border=1><tr><th>Emp No.
                  </th><th>Name</th><th>Designation</th><th>Sal</th></tr>";
      foreach($xml->children() as $books)
      {
         echo "<tr><td>".$books->empno . "</td> ";
         echo "<td>".$books->empname . "</td> ";
         echo "<td>".$books->salary . "</td>";
         echo "<td>".$books->designation. "</td></tr>";
      }
      echo "</table>"
   ?>
```

Output:



Emp No.	Name	Designation	Sal
111	John	10000	Accountant
120	Dane	15000	Asst. Manager
122	Wiely	17000	Manager
100	Robert	12000	Sr. Accountant

Program 2: Program to read item-XMl file (contain INo, Iname, I-desc, Price) and print item details in tabular format (use simple XML).

Item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<product>
   <item>
      <ino>1</ino>
      <iname>Dove</iname>
      <desc>Soap</desc>
      <iprice>55</iprice>
   </item>
   <item>
      <ino>2</ino>
      <iname>Iphone 6</iname>
      <desc>Mobile</desc>
      <iprice>65000</iprice>
   </item>
   <item>
      <ino>3</ino>
      <iname>T-Shirt</iname>
      <desc>Apparel</desc>
      <iprice>450</iprice>
   </item>
   <item>
      <ino>4</ino>
      <iname>Timex</iname>
      <desc>Wrist Watch</desc>
      <iprice>2500</iprice>
   </item>
</product>
```

XML.html

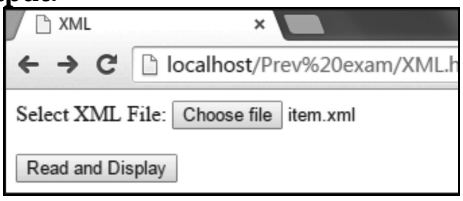
```
<html>
   <head><title>XML</title></head>
   <body>
      <form action="xml.php" method="get">
         Select XML File:
```

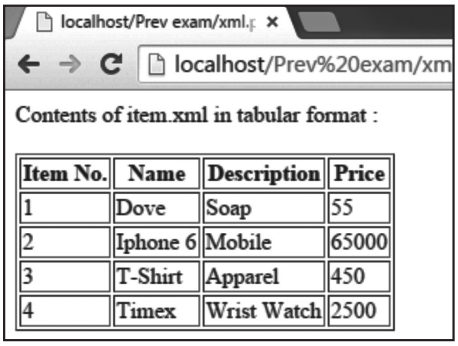
```
            <input type="file" name="fname" />
            <br><br>
            <input type="submit" name="submit" value="Read and Display">
         </form>
      </body>
   </html>
```

Xml.php

```
<?php
   $fname=$_GET['fname'];
   $xml=simplexml_load_file($fname) or die ("Error: Cannot create object");
   echo "Contents of $fname in tabular format : <br><br>";
   echo"<table border=1><tr><th>Item No.</th><th>Name</th><th>
                                        Description</th><th>Price</th></tr>";
   foreach($xml->children() as $books)
   {
      echo "<tr><td>".$books->ino . "</td> ";
      echo "<td>".$books->iname . "</td> ";
      echo "<td>".$books->desc . "</td>";
      echo "<td>".$books->iprice . "</td></tr>";
   }
   echo "</table>"
?>
```

Output:





Item No.	Name	Description	Price
1	Dove	Soap	55
2	Iphone 6	Mobile	65000
3	T-Shirt	Apparel	450
4	Timex	Wrist Watch	2500

Program 3: Program to read book.XML and print book details in tabular format using simple XML. (Content of book.XML are (bookcode, bookname, author, publisher, price).

Books.xml

```
<?xml version="1.0" encoding="utf-8"?>
<bookstore>
   <book category="COOKING">
      <code >02011</code>
      <name lang="en">Everyday Italian</name>
      <author>Giada De Laurentiis</author>
      <year>2005</year>
      <price>30.00</price>
   </book>
   <book category="CHILDREN">
      <code >0156</code>
      <name lang="en">Harry Potter</name>
      <author>J K. Rowling</author>
      <year>2005</year>
      <price>29.99</price>
   </book>
   <book category="WEB">
      <code >2017</code>
      <name lang="en-us">XQuery Kick Start</name>
      <author>James McGovern</author>
      <year>2003</year>
      <price>49.99</price>
   </book>
   <book category="WEB">
      <code >5397</code>
      <name lang="en-us">Learning XML</name>
      <author>Erik T. Ray</author>
      <year>2003</year>
      <price>39.95</price>
   </book>
</bookstore>
```

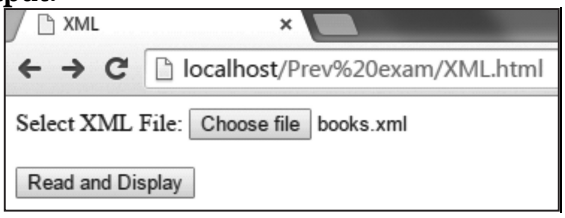
XML.html

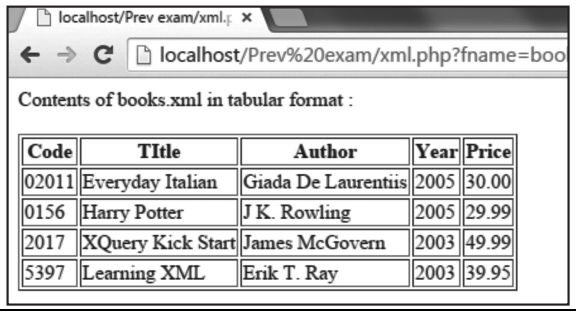
```
<html>
   <head><title>XML</title></head>
   <body>
   <form action="xml.php" method="get">
      Select XML File:
      <input type="file" name="fname" />
      <br><br>
      <input type="submit" name="submit" value="Read and Display">
   </form>
   </body>
</html>
```

Xml.php

```
<?php
   $fname=$_GET['fname'];
   $xml=simplexml_load_file($fname) or die("Error: Cannot create object");
   echo "Contents of $fname in tabular format : <br><br>";
   echo"<table border=1><tr><th>Code</th><th>TItle</th><th>Author
                                    </th><th>Year</th><th>Price</th></tr>";
   foreach($xml->children() as $books)
   {
      echo "<tr><td>".$books->code . "</td> ";
      echo "<td>".$books->name . "</td> ";
      echo "<td>".$books->author . "</td>";
      echo "<td>".$books->year . "</td>";
      echo "<td>".$books->price . "</td></tr>";
   }
   echo "</table>"
?>
```

Output:





Code	TItle	Author	Year	Price
02011	Everyday Italian	Giada De Laurentiis	2005	30.00
0156	Harry Potter	J K. Rowling	2005	29.99
2017	XQuery Kick Start	James McGovern	2003	49.99
5397	Learning XML	Erik T. Ray	2003	39.95

Program 4: Program to read student.xml file which contains student roll no, name, address, college and course. Print student's details of specific course in tabular format after accepting course as input.

Student.xml

```
<?xml version="1.0" encoding="utf-8"?>
<college>
<student>
   <rno>1</rno>
   <name>John</name>
   <addr>PUNE</addr>
   <collegename>Science College</collegename>
   <course>BSC CS</course>
</student>
<student>
   <rno>2</rno>
   <name>Bob</name>
   <addr>Mumbai</addr>
   <collegename>Arts College</collegename>
   <course>BA</course>
</student>
<student>
   <rno>3</rno>
   <name>Wiely</name>
   <addr>Nashik</addr>
   <collegename>Commerce</collegename>
   <course>BCOM</course>
</student>
<student>
```

```
        <rno>4</rno>
        <name>Sam</name>
        <addr>Satara</addr>
        <collegename>Management College</collegename>
        <course>BCA</course>
    </student>
    </college>
```

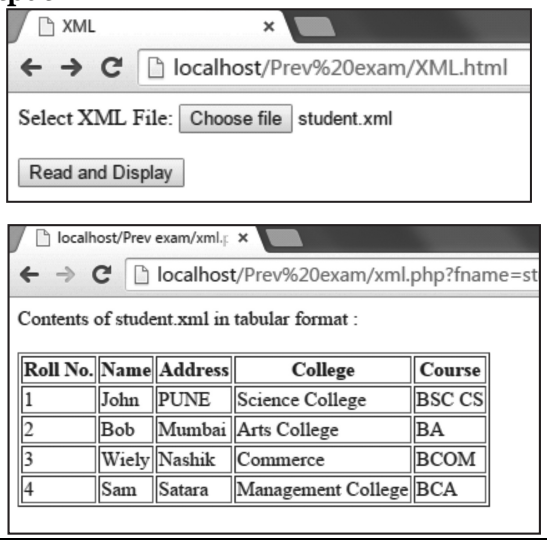
XML.html

```
    <html>
        <head><title>XML</title></head>
        <body>
            <form action="xml.php" method="get">
                Select XML File:
                <input type="file" name="fname" />
                <br><br>
                <input type="submit" name="submit" value="Read and Display">
            </form>
        </body>
    </html>
```

Xml.php

```
    <?php
        $fname=$_GET['fname'];
        $xml=simplexml_load_file($fname) or die ("Error: Cannot create object");
        echo "Contents of $fname in tabular format : <br><br>";
        echo"<table border=1><tr><th>Roll No. </th> <th> Name </th> <th>
                            Address </th> <th> College</th><th>Course</th></tr>";
        foreach($xml->children() as $books)
        {
            echo "<tr><td>".$books->rno . "</td> ";
            echo "<td>".$books->name . "</td> ";
            echo "<td>".$books->addr . "</td>";
            echo "<td>".$books->collegename . "</td>";
            echo "<td>".$books->course . "</td></tr>";
        }
        echo "</table>"
    ?>
```

Output:



Program 5: Program 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.

Account.xml

```
<?xml version="1.0" encoding="utf-8"?>
<bank>
<account>
   <acno>1111</acno>
   <name>bob</name>
   <addr>mumbai</addr>
   <branch>dadar</branch>
   <actype>savings</actype>
   <bal>10000</bal>
</account>
<account>
   <acno>2222</acno>
   <name>sam</name>
   <addr>satara</addr>
   <branch>cbs</branch>
   <actype>current</actype>
   <bal>50000</bal>
```

```
</account>
<account>
   <acno>3333</acno>
   <name>wiely</name>
   <addr>nashik</addr>
   <branch>mg road</branch>
   <actype>current</actype>
   <bal>76000</bal>
</account>
<account>
   <acno>4444</acno>
   <name>john</name>
   <addr>pune</addr>
   <branch>jm road</branch>
   <actype>savings</actype>
   <bal>16000</bal>
</account>
</bank>
```

XML.html

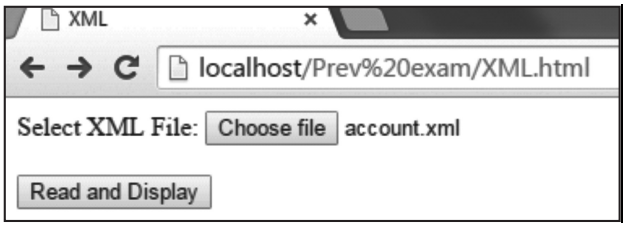
```
<html>
   <head><title>XML</title></head>
   <body>
      <form action="xml.php" method="get">
         Select XML File:
         <input type="file" name="fname" />
         <br><br>
         <input type="submit" name="submit" value="Read and Display">
      </form>
   </body>
</html>
```

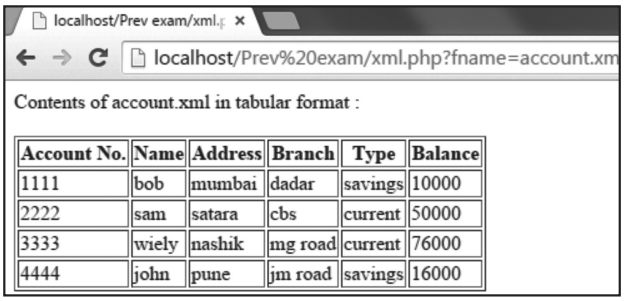
Xml.php

```
<?php
   $fname=$_GET['fname'];
   $xml=simplexml_load_file($fname) or die("Error: Cannot create object");
   echo "Contents of $fname in tabular format : <br><br>";
   echo"<table border=1><tr><th>Account No.</th><th>Name</th><th>
            Address</th><th>Branch</th><th>Type</th><th>Balance</th></tr>";
```

```
    for each($xml->children() as $books)
    {
        echo "<tr><td>".$books->acno . "</td> ";
        echo "<td>".$books->name . "</td> ";
        echo "<td>".$books->addr . "</td>";
        echo "<td>".$books->branch . "</td>";
        echo "<td>".$books->actype . "</td>";
        echo "<td>".$books->bal . "</td></tr>";
    }
    echo "</table>"
?>
```

Output:





Account No.	Name	Address	Branch	Type	Balance
1111	bob	mumbai	dadar	savings	10000
2222	sam	satara	cbs	current	50000
3333	wiely	nashik	mg road	current	76000
4444	john	pune	jm road	savings	16000

PRACTICE QUESTIONS

Q. I Multiple Choice Questions:

1. XML stands for,
   (a) eXtensible Markup Language) (b) eXtensible Makeup Language
   (c) XML eXtensible Machine Language) (d) eXpresentive Markweb Language
2. Following which important characteristics of XML that makes it useful in a variety of systems and solutions.
   (a) XML is extensible (b) XML carries data, does not present it
   (c) XML is a public standard (d) All of the mentioned
3. The XML declaration begin with,
   (a) <xmlns> (b) <?xml>
   (c) <xmln> (d) None of the mentioned
4. CDATA means,
   (a) Computer Data (b) Coding data
   (c) Character Data (d) None of the mentioned
5. The goal of a parser is to transform XML into a readable code. The commonly used parsers are,
   (a) MSXML (Microsoft Core XML Services) (b) System.Xml.XmlDocument
   (c) Java built-in parser (d) All of the mentioned
6. In XML which defines a standard way for accessing and manipulating XML documents and presenting an XML document as a tree-structure.
   (a) Webpage Object Model (WOM) (b) Document Object Model (DOM)
   (c) DocumentTree Object Model (DTOM) (d) None of the mentioned
7. Which is a PHP extension that allows us to easily manipulate and get XML data.
   (a) SimpleXML (b) SimpleXMLN
   (c) SimpleXHTML (d) All of the mentioned
8. Which functions are used to converting SimpleXMl and DOM objects?
   (a) simplexml_import_dom(node) (b) dom_import_simplexml(element)
   (c) Both (a) and (b) (d) None of the mentioned
9. In XML which identify the data and are used to store and organize the data, rather than specifying how to display it?
   (a) tags (b) attributes
   (c) values (d) None of the mentioned

10. Which in XML allow documents to contain instructions for applications?
    (a) Processing Commands (PCs) (b) Processing Instructions (PIs)
    (c) Both (a) and (b) (d) None of the mentioned
11. Following which XML function creates new parser,
    (a) xml_parse() (b) xml_parser_init()
    (c) xml_parser_create() (d) xml_parser_generate()
12. The XML DOM defines the objects, properties and methods of ,
    (a) all XML Classes (b) all XML XSD
    (c) all XML DTDs (d) all XML elements

Answers

1. (a)	2. (d)	3. (b)	4. (c)	5. (d)	6. (b)	7. (a)	8. (c)	9. (a)	10. (b)
11. (c)	12. (d)								

Q. II Fill in the Blanks:

1. XML is a text-based markup language.
2. An XML File is structured by several XML elements also called XML nodes or XML-tags.
3. DOM document stores an entire XML document in the form of a DOM tree.
4. An _____ specifies a single property for the element, using a name/value pair.
5. An XML declartion document is a basic unit of XML information composed of elements and other markup in an orderly package.
6. XML contains details that prepare an XML processor to xml parser the XML document.
7. XML elements can behave as containers to hold text, elements, attributes, media objects or all of these.
8. In XML any text between <--> characters is considered as a comment.
9. XML parser is a software library or a package that provides interface for client applications to work with XML documents.
10. CDATA is defined as blocks of text that are not parsed by the parser, but are otherwise recognized as markup.
11. Processing Instructions (PIs) can be used to pass information to applications and can appear _____ in the document outside the markup.
12. validATION is a process by which an XML document is validated. An XML document is said to be valid if its contents match with the elements, attributes and associated DTD.
13. sIMPLE XML is a PHP extension that allows us to easily manipulate and get XML data.

14. The tree structure is often referred to as Xml Tree. The tree structure contains root (parent) elements, child elements and so on.
15. A DOM document is a collection of nodes (informational units) of information organized in a hierarchy.
16. xml_parse() method start parsing XML document.
17. The simplexml_import_dom (node) method converts the supplied DOM node, node, into a SimpleXMLElement Object.
18. SimpleXML is a Tree -based parser.

Answers

1. XML	2. file	3. DOM	4. attribute
5. declaration	6. parse	7. elements	8. <!-- and -->
9. parser	10. CDATA	11. anywhere	12. Validation
13. SimpleXML	14. XML Tree	15. nodes	16. xml_parse()
17. object	18. tree		

Q. III State True or False:

1. XML allows us to create our own self-descriptive tags or language that suits the application.
2. A well - formed document does not need to be valid, but a valid document must be well – formed.
3. XML elements can have attributes in name/value pairs like HTML.
4. XML can work behind the scene to simplify the creation of HTML documents for large Web sites.
5. Attribute gives more information about XML elements.
6. The XML parser checks for proper format of the XML document and may also validate the XML documents.
7. XML not allows us to store the data irrespective of how it will be presented. F
8. PIs are not part of the character data of the document, but MUST be passed through to the application.
9. DTDs check vocabulary and validity of the structure of XML documents against grammatical rules of appropriate XML language.
10. The XML Schema Definition (XSD) is used to describe and validate the structure and the content of XML data. XML schema defines the elements, attributes and data types.

11. The program creates an XML document with only version line <?xml version="1.0" encoding="UTF-8"?>
12. The XML DOM provides an API that allows a developer to add, edit, move or remove nodes in the tree at any point in order to create an application.
13. The names of XML-elements are case-sensitive. F
14. The SimpleXML PHP extension provides a complete toolset which we can use to read, write and parse XML documents in the PHP applications.
15. XML and PHP are combined to build Web applications. PHP uses set of XML extensions and functions to transform XML mark-up and traverse XML document so as to share data between Web applications.
16. An XML document can also be described as a tree structure.
17. The CDATA are defined as blocks of text that are parsed by the parser. F
18. An XML document is said to be valid if its contents match with the elements, attributes and associated document content if you know the XML document's structure or layout.
19. SimpleXML provides an easy way of getting an element's name, attributes and so on.
20. A PI starts with a special tag <? and ends with ?>.

Answers

1. (T)	2. (T)	3. (T)	4. (T)	5. (T)	6. (T)	7. (F)	8. (T)	9. (T)	10. (T)
11. (T)	12. (T)	13. (F)	14. (T)	15. (T)	16. (T)	17. (F)	18. (T)	19. (T)	20. (T)

Q. IV Answer the following Questions:

(A) Short Answer Questions:

1. What is XML?
2. What is SimpleXML?
3. List major parts of XML.
4. Give any two syntax rule for XML.
5. What is CDATA?
6. What is structure of XML?
7. Give relationship between XML and PHP.
8. Define XML parser.
9. Define DOM.
10. Enlist part of XML document.

(B) Long Answer Questions:

1. What are the characteristics of XML? Explain in detail.
2. Explain document structure of XML.
3. How to process XML? Describe with example.
4. Write short note on: Predefined character entities.
5. Describe CDATA with the example.
6. With example describe XML and PHP.
7. What XML parser? What are its types? Also compare them.
8. What is DOM? How to create XML document using DOM?
9. Write a short note on: Converting SimpleXMl and DOM objects.
10. Describe SimpleXML with example.
11. How to create a new XML parser? Explain with example.
12. What are the applications of XML?

UNIVERSITY QUESTIONS AND ANSWERS

April 2016

1. List the applications of XML. **[1 M]**

Ans. Refer to Section 2.1.

2. XML is case sensitive. Justify true or false. **[1 M]**

Ans. Refer to Section 2.2.2, Point (4).

3. Write PHP script bank.xml file contains bank name, MICR code, IFSC code, address etc. By using XML print bank data in tabular format. **[5 M]**

Ans. Refer to Additional Programs.

April 2017

1. State True/False: "We can have empty XML tags." **[1 M]**

Ans. Refer to Section 2.2.2.

2. Whether root element is required for XML file? If so, how many root elements are required? **[1 M]**

Ans. Refer to Section 2.1.

3. Write a PHP script using simple.xml functions to read student.xml file and display information of students whose percentage is greater than 70. The student.xml file contains name, class and percentage. **[5 M]**

Ans. Refer to Additional Programs.

4. What is well formed XML document? **[2 M]**

Ans. Refer to Section 2.2.

5. What is an XML schema? **[2 M]**

Ans. Refer to Section 2.2.

October 2017

1. What is XML parser? **[1 M]**

Ans. Refer to Section 2.4.

2. What is DOM? **[1 M]**

Ans. Refer to Section 2.5.

3. Explain advantages of XML over HTML? **[5 M]**

Ans. Refer to Section 2.1.

4. Write a script to create XML file 'breakfast .xmls'. The element details of breakfast .xml are as follows:

```
<breakfast menu>
   <food>
      <name>  ……  </name>
      <price>  ……  </price>
      <description>  ……  </description>
      <calories>  ……  </calories>
   <food>
<breakfast menu>
```

Link breakfast .xml file to breakfast .CSS and Get well formatted output as:

Name, Price → Color; Red; Font-family : Arial; Font-size : 15 pt;

Description, calories → Color : Blue; Font-family : Bodoni MT; font-size 12 pt; **[5 M]**

Ans. Refer to Additional Programs.

April 2018

1. What is XML namespaces? **[1 M]**

Ans. Refer to Section 2.2.2.

2. What is use of XML? **[1 M]**

Ans. Refer to Section 2.1.

3. Write PHP script to read book.xml file which contain book number, name of book, name of author, publisher, price. Print book details of specific author in tabular format after accepting name of author as input. **[5 M]**

Ans. Refer to Additional Programs.

October 2018

1. Write any two applications of XML. **[1 M]**

Ans. Refer to Section 2.1.

2. Write any five advantages of XML over HTML. **[5 M]**

Ans. Refer to Section 2.1.

3. Write a PHP script to read account .XML which contains Account No, Name, Address, Branch, Account Type, Balance. Print account details of specific branch in tabular format after accepting branch as input. **[5 M]**

Ans. Refer to Additional Programs.

April 2019

1. Give names of XML parser. **[1 M]**

Ans. Refer to Section 2.4.

2. Write any two applications of XML. **[1 M]**

Ans. Refer to Section 2.1.

3. Write PHP script to read emp.xml file (contains empno, emp_name, salary, designation) and print employee details in tabular format (use SimpleXML). **[5 M]**

Ans. Refer to Additional Programs.

4. Differentiate between XML and HTML. **[4 M]**

Ans. Refer to Section 2.1.

■■■

CHAPTER 3

JavaScript and jQuery

Objectives...

- To study Basic Concepts of JavaScript
- To study JavaScript Control Statements and Loop Statements
- To understand JavaScript Functions and Strings
- To learn JavaScript Pop-up Boxes and Events
- To learn Basic Concepts in jQuery

3.0 INTRODUCTION

- JavaScript is a light-weight object-oriented programming language which is used by several websites for scripting the Web pages.
- JavaScript is an interpreted programming language that enables dynamic interactivity on websites when applied to an HTML document. JavaScript is a dynamic computer programming language.
- JavaScript (JS) is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.
- jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.
- The purpose of jQuery is to make it much easier to use JavaScript on the website.

3.1 OVERVIEW OF JAVASCRIPT AND JQUERY

- JavaScript is a scripting language designed primarily for adding interactivity to Web pages and creating Web applications.
- Script means list of actions for the something to perform. A scripting language is a language that interprets scripts at runtime. Scripts are usually embedded into other software environments.
- Scripting languages are of two types namely, Client-side scripting languages and Server-side scripting languages.

- Client-server script runs at user's end i.e. the browser execute the scripts. JavaScript is an example of a client side script that is interpreted by the client browser.
- Where as server-side scripting language is executed on a web server. PHP is a server side script that is interpreted on the server.

3.1.1 JavaScript

- JavaScript is a lightweight, interpreted programming language.
- JavaScript is very easy to implement because it is integrated with HTML. It is open and cross-platform.
- JavaScript most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

3.1.1.1 Features of JavaScript

- Features of JavaScript are listed below:
  1. JavaScript is an object-based scripting language.
  2. It is light weighted.
  3. JavaScript is a scripting language and it is not java.
  4. JavaScript is interpreter based scripting language.
  5. JavaScript gives HTML designers a programming tool.
  6. JavaScript can put dynamic text into an HTML page.
  7. JavaScript can react to event.
  8. JavaScript can read and write HTML document.
  9. JavaScript can be used to validate data.
  10. JavaScript is object based language as it provides predefined objects.
  11. Most of the JavaScript control statements syntax is same as syntax of control statements in C language.

3.1.1.2 Advantages and Disadvantages of JavaScript [April 18, 19]

- Various advantages of JavaScript are given below: [April 19]
  1. **Less server interaction:** We can validate user input before sending the page to the server, means less load on the server.
  2. **Immediate feedback to the visitors:** They do not have to wait for a page reload to see if they have forgotten to enter something.
  3. **Increased interactivity:** We can create interfaces that react when the user put mouse on that interface.
  4. **Richer interfaces:** We can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to our site visitors.

5. **Speed:** Being client-side, JavaScript is very fast because any code functions can be run immediately instead of having to contact the server and wait for an answer.
6. **Simplicity:** JavaScript is relatively simple to learn and implement.
7. **Versatility:** JavaScript plays nicely with other languages and can be used in a huge variety of applications. Unlike PHP or SSI scripts, JavaScript can be inserted into any web page regardless of the file extension. JavaScript can also be used inside scripts written in other languages such as Perl and PHP.
8. **Server load:** Being client-side reduces the demand on the website server.

- Various disadvantages of JavaScript are given below: **[April 18]**

1. Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
2. JavaScript cannot be used for networking applications because there is no such support available.
3. JavaScript does not have any multithreading or multiprocessing capabilities.
4. **Security:** Because the code executes on the users' computer, in some cases it can be exploited for malicious purposes. This is one reason some people choose to disable JavaScript.
5. **Reliance on End User:** JavaScript is sometimes interpreted differently by different browsers. Whereas, server-side scripts will always produce the same output, client-side scripts can be a little unpredictable. Don't be overly concerned by this though - as long as you test your script in all the major browsers you should be safe.

3.1.2 JQuery

- jQuery is a lightweight, "write less, do more", JavaScript library.
- jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation as well as event handling, CSS animation and Ajax.
- jQuery is free, open-source software using the permissive MIT License. The MIT License is a permissive free software license originating at the Massachusetts Institute of Technology (MIT) in the late 1980s.
- jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications.
- jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library.
- This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, theme-able widgets.
- jQuery is a fast, small, cross-platform and feature-rich JavaScript library. It is designed to simplify the client-side scripting of HTML.

- The main purpose of jQuery is to provide an easy way to use JavaScript on your website to make it more interactive and attractive. It is also used to add animation.
- JQuery is also very useful to simplify a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

Features supported by jQuery:

1. **Lightweight:** The jQuery is very lightweight library - about 19KB in size (minified and gzipped).
2. **DOM Manipulation:** The jQuery made it easy to select DOM elements, negotiate them and modifying their content by using cross-browser open source selector engine called Sizzle.
3. **Event Handling:** The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.
4. **Latest Technology:** The jQuery supports CSS3 selectors and basic XPath syntax.
5. **AJAX Support**: The jQuery helps us, a lot to develop a responsive and feature rich site using AJAX technology.
6. **Animations**: The jQuery comes with plenty of built-in animation effects which we can use in the websites.
7. **Cross Browser Support**: The jQuery has cross-browser support and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+.

Advantages of jQuery:

1. **Simple and Easy:** JavaScript have predefined methods using we can perform any task easily compare to JavaScript. jQuery is easy to learn.
2. **Cross browser support:** jQuery support all modern web-browser.
3. **Lightweight:** jQuery is very lightweight library.
4. **CSS Manipulation:** jQuery have predefined css() method for manipulate style for any HTML elements.
5. **HTML Manipulation:** The jQuery made it easy to select DOM elements, traverse them and modifying their content.
6. **Extendibility:** We start out with a core library and extend its functionality per our needs with plugins.

Disadvantages of jQuery:

1. jQuery Not built for larger and complex applications.
2. For larger and more complicated applications we will need to extend way beyond core with additional libraries and Plugins.
3. jQuery is easy to install and learn, initially. But it's not that easy if we compare it with CSS.

3.2 OBJECT ORIENTATION AND JAVASCRIPT

- JavaScript features powerful, flexible OOP capabilities. In JavaScript, you can write code that can be re-used and that is encapsulated.
- JavaScript is all about objects. Windows and buttons, forms and images, links and anchors are all objects.
- Programming languages like Java, C++ and Python that focus on objects are called Object-Oriented Programming (OOP) languages.
- JavaScript is called an object-based language because it doesn't technically meet the criteria of the more heavy-duty languages, but it certainly behaves as an object-oriented language.
- In the real world, we may see a book or a car as an object. JavaScript can represent data such as a string or a number as an object and JavaScript lets we create our own objects also.
- Object-oriented languages, such as C++ and Java, bundle up data and behaviour and call it an object. So does JavaScript.
- JavaScript sees the browser window as an object, a window that can be resized, opened, closed, and so on.
- It sees all the frames, documents, images, and widgets inside the window as objects. And these objects have properties and methods.
- JavaScript supports several types of objects, as follows:
  1. User-defined objects defined by the programmer.
  2. Core or built-in objects, such as Date, String, and Number.
  3. Browser objects, the BOM.
  4. The Document objects, the DOM.
- Core objects are built right into the language. JavaScript provides built-in objects that deal with the date and time, math, strings, regular expressions, numbers, and other useful entities.
- The core objects are consistent across different implementations and platforms, and were standardized by the ECMAScript 1.0 specification, allowing programs to be portable.
- JavaScript programs are associated with a browser window and the document displayed in the window. The window is a browser object and the document is an HTML object.
- In the browser object model, sometimes called BOM, the window is at the top of the tree, and below it are objects: window, navigator, frames[], document, history, location and screen.

3.3 BASIC SYNTAX (JS DATATYPES AND JS VARIABLES)

- JavaScript is case sensitive language.
- Two types of comments used in JavaScript are given below:
  1. Multi line comment,

```
/* ---------------------
      ----------------- */
```

  2. Single line comment,

```
// ----------------------
```

- The { } are used to indicate a block of code and ; is optional at the end of the statement.
- JavaScript can be implemented using JavaScript statements that are placed within the <script> ... </script> HTML tags in a web page.
- We can place the <script> tags, containing your JavaScript, anywhere within the Web page, but it is normally recommended that you should keep it within the <head> tags.
- The <script> tag alerts the browser program to start interpreting all the text between these tags as a script.
- A simple syntax of your JavaScript will appear as follows:

```
<script ...>
   JavaScript code ...
</script>
```

- The script tag takes two important attributes:
  1. **Language:** This attribute specifies what scripting language you are using. Typically, its value will be JavaScript.
  2. **Type:** This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".
- So your JavaScript segment will look like:

```
<script language="javascript" type="text/javascript">
   JavaScript code ...
</script>
```

- The <script> and </script> informs browser that everything in between is a scripting language. The attribute type, is given a value "text/javascript", specifying that the scripting language is javascript.

Embedding JavaScript:

- JavaScript can be added in the <body> or in the <head> sections of an HTML code.

1. **Example of Embedding JavaScript in <head> Tag:**

```
<html>
   <head>
      <title>Javascript Example </title>
      <script type="text/javascript">
```

```
            document.write("My First Javascript");
        </script>
    </head>
</html>
```

Output:

```
My first Javascript
```

- In document.write document is the part of browser where we see webpage content, document.write will write the text in parentheses (round brackets) to the browser document.
- In this case, the text "My First Javascript" will be written. The double quotes are important when we want to print/write the text in is. And every statement in JavaScript ends with a semi-colon (;).

2. Example shows Embedded JavaScript in <body > Tag:

```
<html>
    <body>
        This is body.
        <script type="text/javascript">
            document.write("Hello World!");
        </script>
    </body>
</html>
```

Output:

```
This is body.
Hello World!
```

3. Embedding a JavaScript using External File: **[April 17]**

- First create “myScript.js” file write the following JavaScript code in it.

```
document.write("This is JavaScript");
```

- Now create an HTML file and put the following code. In this HTML the JavaScript is embedded externally.

```
<html>
    <body>
        This is body <br>
        <script type="text/javascript" src="myScript.js"></script>
    </body>
</html>
```

Output:

```
This is body
This is JavaScript
```

3.3.1 JavaScript Data Types [April 16, 19]

- One of the most fundamental characteristics of a programming language is the set of data types it supports. These are the type of values that can be represented and manipulated in a programming language.
- JavaScript provides different data types to hold different types of values. There are two types of data types in JavaScript.
  1. Primitive data type, and
  2. Non-primitive (Reference) data type.
- There are five types of primitive data types in JavaScript. They are as follows:

Data Type	Description
String	Represents sequence of characters e.g. "hello".
Number	Represents numeric values e.g. 100.
Boolean	Represents boolean value either false or true.
Undefined	Represents undefined value.
Null	Represents null i.e. no value at all.

- The non-primitive data types are as follows:

Data Type	Description
Object	Represents instance through which we can access members.
Array	Represents group of similar values.
RegExp	Represents regular expression.

3.3.2 Regular Expression

- A regular expression is a sequence of characters that forms a search pattern.
- When we search for data in a text, we can use this search pattern to describe what we are searching for.
- A regular expression can be a single character, or a more complicated pattern.
- Regular expressions can be used to perform all types of text search and text replace operations.

  Syntax: `/pattern/modifiers;`

  Example: `var patt = /niraliprakashan/i;`

  Example explained:

  `/ niraliprakashan/i` is a regular expression.

  `niraliprakashan` is a pattern (to be used in a search).

  `i` is a modifier (modifies the search to be case-insensitive).

3.3.3 Variables [April 17]

- Variables are "containers" for storing information.
- JavaScript variables are used to hold values (a = 5) or expressions (x = y + z) in them.
  Syntax: `var variable_name = value;`
- We define variables in JavaScript using a keyword var. variable_name is name of the variable which holds some value.
  Example: `var x = 10;`
- In the above statement, we assigned value "10" to a variable x.
- There are some rules for variable names in JavaScript:
  1. A variable must start with an alphabet or an underscore.
  2. The subsequent characters can be digits (0-9) or alphabets or an underscore.
  3. As JavaScript is case-sensitive, number is not same as Number.
- Some valid examples of variable names in JavaScript are given below:
  `simple_Interest, _name, number1`
- Some invalid examples of variable names are given below:
  `23Amar, &MyName, Mega%tron`

Variable Scope: [April 17]

- The scope of a variable is the region of our program in which it is defined.
- JavaScript variable will have only two scopes:
  1. **Global variables:** A global variable has global scope which means it is defined everywhere in our JavaScript code.
  2. **Local variables:** A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.
- Global variables are accessible from anywhere in the program. On other hand, Local variables are accessible only in the function they are defined.

Example:

```
<html>
      <head>
      <title>Javascript Local and Global Variables Example</title>
      </head>
      <body>
         <script type="text/javascript">
            var a = 10; //global variable
            function test()
            {
               document.write("a = "+a); //access inside function
               var b = 20; // local variable
```

```
                document.write("<br/>");
                document.write("b = "+b); //access inside function
                document.write("<br/>");
            }
            test();
                document.write("a = "+a); //acces outside function
                document.write("b = "+b); //not acces outside function
        </script>
    </body>
</html>
```

Output:

```
a = 10
b = 20
a = 10
```

- We declared a=10; outside the function, hence its a global variable, and can be accessed from anywhere in the program.
- In the above example, we accessed it from inside the function and as well as outside the function using document.write("a = "+a); statement.
- We declared b=20; inside the function, hence it's a local variable, and can be accessed only within the function but from the outside the function it is not accessible.

3.4 PRIMITIVES, OPERATIONS AND EXPRESSIONS

[Oct. 17, April 18]

- An operator is used to transform one or more values into a single resultant values and the values to which the operator is applied is referred as operands.
- JavaScript language supports following type of operators:
  1. Arithmetic operators,
  2. Comparison operators,
  3. Logical (or Relational) operators,
  4. Assignment operators, and
  5. Conditional (or ternary) operators.
- Let us see above operators in detail:

1. Arithmetic Operators:

- Arithmetic operators are used to perform arithmetic between variables and/or values.

- Following table shows arithmetic operators supported by JavaScript language. Assume variable A holds 10 and variable B holds 20 then:

Sr. No.	Operator	Description	Example
1.	+ (Addition Operator)	Adds two operands.	A + B = 30
2.	– (Subtraction Operator)	Subtracts second operand from the first.	A – B = –10
3.	* (Multiplication Operator)	Multiply both operands.	A * B = 200
4.	/ (Division Operator)	Divide numerator by denumerator.	B / A = 2
5.	% (Modulus Operator)	Modulus operator and remainder of after an integer division.	B % A = 0
6.	++ (Increment Operator)	Increases integer value by one.	A++ = 11
7.	-- (Decrement Operator)	Decreases integer value by one.	A-- = 9

2. Comparison Operators:

- Comparison operators are used to compare a condition and are always inside conditional statements. A comparison evaluate to their true or false. These true and false values are called Booleans.
- Following table shows comparison operators supported by JavaScript and assume variable A holds 10 and B holds 20.

Sr. No.	Operator	Description	Example
1.	== (Equal To Operator)	This operator checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
2.	!= (Not Equal To Operator)	This operator checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
3.	> (Greater Than Operator)	This operator checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
4.	< (Less Than Operator)	This operator checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.

contd. …

5.	>= (Less Than or Equal To Operator)	This operator checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
6.	<= (Greater Than or Equal To Operator)	This operator checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

3. Logical Operators:

- Using logical operators, we can test two or more conditions in one line code.
- There are following logical operators supported by JavaScript and assume variable A holds 10 and variable B holds 20 then:

Sr. No.	Operator	Description	Example
1.	&& (Logical AND Operator)	If both the operands are non zero then condition becomes true.	(A && B) is true.
2.	|| (Logical OR Operator)	If any of the two operands are non zero then then condition becomes true.	(A || B) is true.
3.	! (Logical NOT Operator)	Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

4. Bitwise Operators:

- Bitwise operators act upon the individual bits of their operands.
- Following table gives bitwise operators supported by JavaScript and assume variable A holds 2 and variable B holds 3 then:

Sr. No.	Operator	Description	Example
1.	& (Bitwise AND operator)	It performs a Boolean AND operation on each bit of its integer arguments.	(A & B) is 2.
2.	| (Bitwise OR Operator)	It performs a Boolean OR operation on each bit of its integer arguments.	(A | B) is 3.

contd. ...

3.	^ (Bitwise XOR Operator)	It performs a Boolean exclusive OR operation on each bit of its integer arguments. Exclusive OR means that either operand one is true or operand two is true, but not both.	(A ^ B) is 1.
4.	~ (Bitwise NOT Operator)	It is a is a unary operator and operates by reversing all bits in the operand.	(~B) is -4 .
5.	<< (Bitwise Shift Left Operator)	It moves all bits in its first operand to the left by the number of places specified in the second operand. New bits are filled with zeros. Shifting a value left by one position is equivalent to multiplying by 2, shifting two positions is equivalent to multiplying by 4, etc.	(A << 1) is 4.
6.	>> (Bitwise Shift Right with Sign Operator)	It moves all bits in its first operand to the right by the number of places specified in the second operand. The bits filled in on the left depend on the sign bit of the original operand, in order to preserve the sign of the result. If the first operand is positive, the result has zeros placed in the high bits; if the first operand is negative, the result has ones placed in the high bits. Shifting a value right one place is equivalent to dividing by 2 (discarding the remainder), shifting right two places is equivalent to integer division by 4, and so on.	(A >> 1) is 1.
7.	>>> (Bitwise Shift Right with Zero Operator)	This operator is just like the >> operator, except that the bits shifted in on the left are always zero.	(A >>> 1) is 1.

5. Assignment Operators:

- Assignment operators are used to assign values to JavaScript variables.
- There are following assignment operators supported by JavaScript.

Sr. No.	Operator	Description	Example
1.	= (Simple Assignment Operator)	Assigns values from right side operands to left side operand.	C = A + B will assignee value of A + B into C.
2.	+= (Add and Assign-ment Operator)	It adds right operand to the left operand and assign the result to left operand.	C += A is equivalent to C = C + A.
3.	-= (Subtract and Assignment Operator)	It subtracts right operand from the left operand and assign the result to left operand.	C -= A is equivalent to C = C - A.
4.	*= (Multiply and Assignment Operator)	It multiplies right operand with the left operand and assign the result to left operand.	C *= A is equivalent to C = C * A.
5.	/= (Divide and Assignment Operator)	It divides left operand with the right operand and assign the result to left operand.	C /= A is equivalent to C = C / A.
6.	%= (Modulus and Assignment Operator)	It takes modulus using two operands and assign the result to left operand.	C %= A is equi-valent to C = C % A.

6. Conditional Operators (?:):

- There is an operator called conditional operator. This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation.
- The conditional operator has following syntax:

Sr. No.	Operator	Description	Example
1.	?:	Conditional Expression	If Condition is true? Then value X: Otherwise value Y.

7. typeof Operator:

- The typeof is a unary operator that is placed before its single operand, which can be of any type. Its value is a string indicating the data type of the operand.
- The typeof operator evaluates to "number", "string", or "boolean" if its operand is a number, string, or boolean value and returns true or false based on the evaluation.

- Here, is the list of return values for the typeof Operator:

Type	String Returned by typeof
Number	"number"
String	"string"
Boolean	"boolean"
Object	"object"
Function	"function"
Undefined	"undefined"
Null	"object"

Expressions:

- An expression is any valid set of literals, variables, operators, and expressions that evaluates to a single value.
- The value may be a number, a string, or a logical value. Conceptually, there are two types of expressions i.e., those that assign a value to a variable, and those that simply have a value.
- For example, the expression x = 7 is an expression that assigns x the value 7. This expression itself evaluates to 7. Such expressions use assignment operators.
- On the other hand, the expression 3 + 4 simply evaluates to 7; it does not perform an assignment. The operators used in such expressions are referred to simply as operators.

3.5 JS CONTROL STATEMENTS [April 19]

- While writing a program, there may be a situation when we need to adopt one path out of the given two paths.
- So we need to make use of conditional statements that allow our program to make correct decisions and perform right actions.
- JavaScript supports conditional statements which are used to perform different actions based on different conditions.
- In JavaScript we have the following conditional statements:
  1. **if Statement:** Use if statement to execute some code only if a specified condition is true.
  2. **if...else Statement:** Use if...else statement to execute some code if the condition is true and another code if the condition is false.
  3. **if...elseif....else Statement:** Use if...elseif...else statement to select one of many blocks of code to be executed.
  4. **switch Statement:** Use switch statement to select one of many blocks of code to be executed.

3.5.1 if Statement

- The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.
- We can use the if statement to execute some code only if a specified condition is true.

Syntax:

```
if(condition)
{
   // execute this statement;
}
```

Example:

```
<!DOCTYPE html>
<html>
   <head><title>If statement</title></head>
   <body>
      <script type="text/javascript">
         var x = 7;
         var y = 7;
         if(x==y)
         {
            document.write("Both are equal");
         }
      </script>
   </body>
</html>
```

Output:

```
Both are equal
```

3.5.2 if...else Statement

- We can use the 'if...else' statement to execute some code if a condition is true and another code if the condition is not true.
- The keyword if executes a statement only if the condition is true. It does not do anything if the condition is false.

Syntax:

```
if(condition)
{
   // execute this statement;
}
```

```
else
{
   // execute this statement;
}
```

Example:

```
<script>
   var a=40;
   if(a%2==0)
   {
      document.write("a is even number");
   }
   else{
      document.write("a is odd number");
   }
</script>
```

Output:

```
a is even number
```

3.5.3 if... elseif... else Statement

- The 'if...else if... statement is the one level advance form of control statement that allows JavaScript to make correct decision out of several conditions.

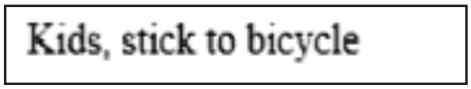
Syntax:

```
if(condition)
{
   // execute this statement;
}
else if(condition)
{
   // execute this statement;
}
else
{
   // execute this statement;
}
```

Example:

```
<!DOCTYPE html>
<html>
   <body>
      <script type="text/javascript">
         var your_age = 14;
         var friends_age = 16;
         if(your_age >= 18)
         {
            document.write("Get a drivers license");
         }
         else if(friends_age >= 18)
         {
            document.write("Let our friend drive the car");
         }
         else
         {
            document.write("Kids, stick to bicycle");
         }
      </script>
   </body>
</html>
```

Output:



3.5.4 switch Statement

- Switch statement is used to execute one of the statements from many blocks of statements.
- Switch statement is like enhanced if-else if-else statement, only less confusing and more easy and simple to use.

Syntax:

```
switch(value/expression)
{
case "value1":
   // execute this statement1;
break;
```

```
case "value2":
   // execute this statement2;
break;
---
---
---
default:
   // executes this statement;
}
```

Example:

```
<!DOCTYPE html>
<html>
   <body>
      <script>
         var x;
         var d=new Date().getDay();
         switch (d)
         {
         case 0:
            x="Today it's Sunday";
         break;
         case 1:
            x="Today it's Monday";
         break;
         case 2:
            x="Today it's Tuesday";
         break;
         case 3:
            x="Today it's Wednesday";
         break;
         case 4:
            x="Today it's Thursday";
         break;
         case 5:
            x="Today it's Friday";
         break;
```

```
            case 6:
               x="Today it's Saturday";
            break;
            default:
               x="Looking forward to the Weekend";
            }
            document.write(x);
         </script>
      </body>
   </html>
```

Output:

```
Today it's Sunday
```

3.5.5 Loops

- It is often the case that we want to do something fixed number of times or until a particular condition has been met. In JavaScript, this repetitive operation is done using loops.
- Loops can execute a block of code a number of times.
- JavaScript supports following kinds of loops:
  1. **for:** Loops through a block of code a number of times.
  2. **for/in:** Loops through the properties of an object.
  3. **while**: Loops through a block of code while a specified condition is true.
  4. **do/while:** Loops through a block of code while a specified condition is true.

3.5.5.1 for Loop

- The 'for' loop is the most compact form of looping. It includes the following three important parts:
  1. **The loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
  2. **The test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
  3. **The iteration statement** where you can increase or decrease your counter.

Syntax:

```
for(initialize; condition; increment)
{
   // execute statements;
}
```

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <script type="text/javascript">
         for(var x=6; x<=10; x++)
         {
            document.write("The number is: " +x);
            document.write("<br/>");
         }
      </script>
   </head>
</html>
```

Output:

The number is: 6
The number is: 7
The number is: 8
The number is: 9
The number is: 10

3.5.5.2 for/in Loop

- JavaScript for/in statement loops through the properties of an object.

Syntax:

```
for(variablename in object)
{
   // statement or block to execute …
}
```

- In each iteration, one property from object is assigned to variablename and this loop continues till all the properties of the object are exhausted.

Example:

```
<html>
   <body>
      <script>
         var x;
         var txt="";
         var person={fname:"John",lname:"Doe",age:25};
```

```
            for(x in person)
            {
               document.write(person[x] + " ");
            }
         </script>
      </body>
   </html>
```

Output:

```
John Doe 25
```

3.5.5.3 while Loop

- The purpose of a while loop is to execute a statement or code block repeatedly as long as an expression is true. Once, the expression becomes false, the loop terminates.

Syntax:

```
initialize;
while (condition)
{
   // execute statement;
   increment;
}
```

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <script type="text/javascript">
         var number = 3;
         while (number <= 10)
         {
            document.write("The number is: " +number);
            document.write("<br/>");
            number++;
         }
      </script>
   </head>
</html>
```

Output:

3.5.5.4 do/while Loop

- Sometimes, we want some statements to be executed atleast once even if the condition is false for the first time. To do this we use a do-while loop.

Syntax:

```
do
{
   // Statements to be executed;
}while (expression);
```

Example:

```
<html>
   <body>
      <script>
         var x="",i=0;
         do
         {
         x=x + "The number is " + i + "<br>";
         i++;
         }
         while (i<5)
            document.write(x);
      </script>
   </body>
</html>
```

Output:

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
```

3.5.6 Loop Control Statements

- JavaScript provides full control to handle loops and switch statements. There may be a situation when you need to come out of a loop without reaching at its bottom.
- There may also be a situation when you want to skip a part of your code block and start the next iteration of the look.
- To handle all such situations, JavaScript provides break and continue statements. These statements are used to immediately come out of any loop or to start the next iteration of any loop respectively.

3.5.6.1 break Statement

- The break statement indicates the end of that particular case. The break statement was used to "jump out" of a switch() statement.
- The break statement can also be used to jump out of a loop. The break statement breaks the loop and continues executing the code after the loop (if any).

Syntax:

```
break;
```

Example:

The following example illustrates the use of a break statement with a while loop. Notice how the loop breaks out early once x reaches 5 and reaches to document.write (..) statement just below to the closing curly brace:

```
<html>
   <body>
      <script type="text/javascript">
         <!--
         var x = 1;
         document.write("Entering the loop<br /> ");
         while (x < 20)
         {
            if (x == 5){
            break; // breaks out of loop completely
            }
            x = x + 1;
            document.write( x + "<br />");
          }
            document.write("Exiting the loop!<br /> ");
         //-->
       </script>
          <p>Set the variable to different value and then try...</p>
   </body>
</html>
```

Output:

```
Entering the loop
   2
   3
   4
   5
Exiting the loop!
Set the variable to different value and then try...
```

- We already have seen the usage of break statement inside a switch statement.

3.5.6.2 continue Statement

- The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

  Syntax: `continue;`

Labels:

- JavaScript statements can be labeled. To label JavaScript statements we proceed the statements with a colon(:).

```
label:
statements
```

- The break and the continue statements are the only JavaScript statements that can "jump out of" a code block.

Syntax:

```
break labelname;
continue labelname;
```

- The continue statement (with or without a label reference) can only be used inside a loop.
- The break statement, without a label reference, can only be used inside a loop or a switch.
- With a label reference, it can be used to "jump out of" any JavaScript code block.

Example:

```
<!DOCTYPE html>
<html>
   <body>
      <script>
         cars=["Ritz","Alto","Bravo","Ford"];
         list:
         {
            document.write(cars[0] + "<br>");
```

```
                document.write(cars[1] + "<br>");
                document.write(cars[2] + "<br>");
                break list;
                document.write(cars[3] + "<br>");
                document.write(cars[4] + "<br>");
                document.write(cars[5] + "<br>");
            }
        </script>
    </body>
</html>
```

Output:

```
Ritz
Alto
Bravo
```

3.6 JS FUNCTIONS

- Like any other advance programming language, JavaScript also supports all the features necessary to write modular code using functions.
- A function is a block of code that will be executed when "someone" calls it. Function is a block of statements that performs certain task.
- Functions are of two types' pre-defined/built-in functions and user-defined functions.
  1. **Built-in functions** are the functions that are already defined in the JavaScript. Examples are write(), alert(), prompt() etc.
  2. **User-defined functions** are defined by a user. Sometimes, this functions are simple, and sometimes they are quite complex.

Function Definition:

- The most common way to define a function in JavaScript is by using the function keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Syntax:

```
<script type="text/javascript">
    function function_name(parameter-list)
    {
        // Statements …
    }
</script>
```

Calling a Function:

- A function can be called from any section of our code.

  Syntax: `functionName()`;

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Javascript Function Example </title>
   </head>
   <body>
      <script type="text/javascript">
         function test()
         { //defining function
            var a = 40;
            document.write(a);
         }
         test();        //calling function
      </script>
   </body>
</html>
```

Output:

```
40
```

- We have defined a function test using keyword function. In the function block we have assigned a value 40 to variable a. The next line is document.write() statement.
- This function test() will not print 40 to the screen all by itself. It has to be called, which we are doing using the statement test();

Function Parameters:

- In the previous point, we defined a function test() without any values in the round brackets.
- In practical situations, we might need to add some values (parameters) in the round brackets.

Syntax:

```
function functionName(para_1, para_2...para_n)
{
   // statements to be executed;
}
```

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Javascript Function Parameters</title>
   </head>
   <body>
      <script type="text/javascript">
         function add(x,y)
         {
            result = x+y;
            document.write("addition is: "+result);
            document.write("<br/>");
         }
         add(10,10);
         add(23,12);
      </script>
   </body>
</html>
```

Output:

addition is: 20
addition is: 35

return Statement:

- A JavaScript function can have an optional return statement. This is required if we want to return a value from a function. This statement should be the last statement in a function.
- For example, we can pass two numbers in a function and then we can expect from the function to return their multiplication in our calling program.

  Syntax: `return value;`

Example:

```
<!DOCTYPE html>
<html>
   head>
      <title>Javascript Return statement</title>
   </head>
```

```
    <body>
        <script type="text/javascript">
            function areaRect(L,B)
            {
                var area = L*B;
                return area;
            }
            x=areaRect(6,8);
            document.write(x);
        </script>
    </body>
</html>
```

Output:

48

- areaRect(6,8); calls the function and passes values 6 and 8.
- The function calculates the area, and returns it (in our case, 48).
- Now, after the function has returned a value, areaRect(6,8) is 48. This value 48, is assigned to x.
- The statement document.write(x) will print 48 in the webpage.

Example: This program shows a button. After you click on the button, the function show() will be called and it displays the output “Hello World”.

```
<!DOCTYPE html>
<html>
    <head>
        <script type="text/javascript">
            function show()
            {
                document.write("Hello World");
            }
        </script>
    </head>
    <body>
        <form>
            <input type="button" value="Click Me" onclick="show()">
        </form>
    </body>
</html>
```

Output:

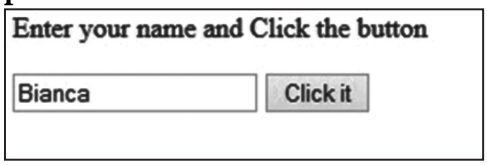
After we click on button displays, `Hello World`.

- This program performs event handling i.e. onclick event.

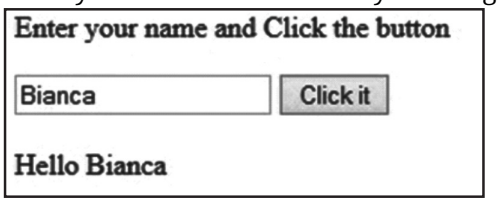
Example:

```
<!DOCTYPE html>
<html>
   <head>
      <script>
         function myFunction()
         {
            var x = "Hello " + document.getElementById("n1").value;
            document.getElementById("demo").innerHTML=x;
         }
      </script>
   </head>
   <body>
      <p>Enter your name and Click the button</p>
      <input type="text" name="name1" id="n1">
      <button onclick="myFunction()">Click it</button>
      <p id="demo"></p>
   </body>
</html>
```

Output:



- After you click on the button you will get the following output:



- In this program displays a form with a text field and a button. After we click on the button function myFunction() will be called the property of the document object getElementById()will get value entered in the text field.
- Another property innerHTML will put the contents of variable x into the place specified by the id “Demo”.

3.7 JAVASCRIPT HTML DOM EVENTS [April 16, 19]

- Every web page resides inside a browser window which can be considered as an object.
- A Document object represents the HTML document that is displayed in that window.
- The Document object has various properties that refer to other objects which allow access to and modification of document content.
- The way that document content is accessed and modified is called the Document Object Model, or DOM.
- The Objects are organized in a hierarchy as shown in Fig. 3.1. This hierarchical structure applies to the organization of objects in a Web document.
  - **Window Object:** Top of the hierarchy. It is the outmost element of the object hierarchy. The window object represents an open window in a browser. An object of window is created automatically by the browser. **[April 16, Oct. 18]**
  - **Navigator Object:** The JavaScript navigator object is the object representation of the client Internet browser or web navigator program that is being used. Navigator object is the top level object to all others. The navigator object contains information about the browser.
  - **Document Object:** Each HTML document that gets loaded into a window becomes a document object. The document contains the content of the page. The document object is itself a property of the window object. This makes sense, because all documents in a Web browser are displayed in a window.
  - **Form Object:** Everything enclosed in the <form>...</form> tags sets the form object.
  - **Form control elements:** The form object contains all the elements defined for that object such as text fields, buttons, radio buttons, and checkboxes.
  - The **History Object**, also a property of the window objects, represents the history list in the window. The history object is automatically created by the JavaScript runtime engine and consists of an array of URLs. These URLs are the URLs the user has visited within a browser window. The history object is part of the Window object and is accessed through the window.history property. The history list contains the most recent pages we have visited during the current browser session, those that we can get to with the Back and Forward buttons.
  - The **Location Object**, also a property of the window, represents the location of the current document (usually shown in the location bar, also known as the address bar) in the browser window. The location object is actually a JavaScript object, not an HTML DOM object. The location object is automatically created by the JavaScript run-time engine and contains information about the current URL. The location object is part of the Window object and is accessed through the window.location property.

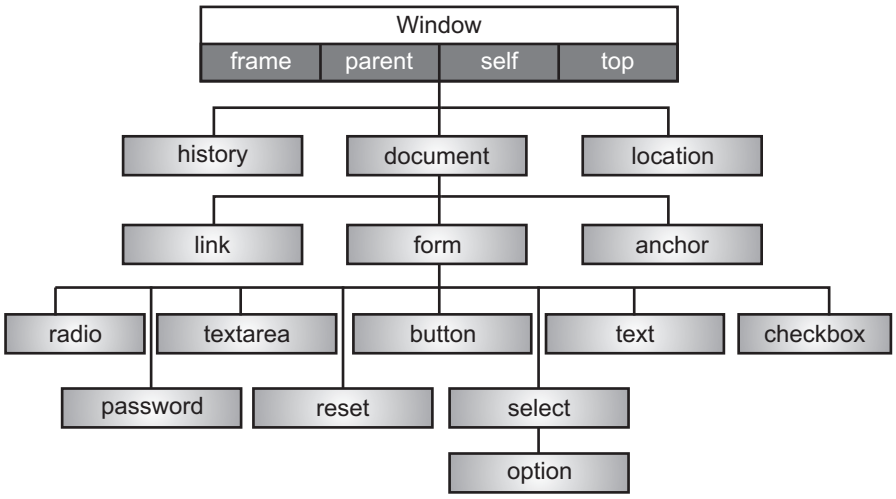


Fig. 3.1

- The navigator object represents the browser and is one of two top-level objects, its name is rather Netscape biased, "browser" would be a non-biased name, but then Netscape was the company that created the JavaScript language, so we think that entitles them to some liberties. The window object is the other top-level object.

JavaScript DOM Objects:

1. **The anchors Array:**

- The anchors array is an array of anchor objects and it is a property of the document object.
- This objects are reflections of a elements that have a name attribute:

```
<a name="anchorName">anchorText</a>
```

2. **The button Object:**

- It is a reflection of an input element with a type attribute of "button".

```
<input type="button">
```

- Button objects are members of the containing form object's elements array.

3. **The checkbox Object:**

- This object is a reflection of an input element with a type attribute of "checkbox".

```
<input type="checkbox">
```

- Checkbox objects are members of the containing form's elements array.

4. **The Document Object:**

- This object contains information about the currently displayed document and its properties are derived from the document's body element.

```
<body>document contents</body>
```

The elements Array:

- It contains references to input elements in a form.
- We cannot add elements to this array, replace elements in the array or remove elements from the array. The elements array is a form object property.

5. The form Object:

- This object collects input from the user and may send the input to a server and it is a reflection of a form element:

```
<form>form contents</form>
```

- Each and every form object is a member of the containing document object's forms array.

The forms Array:

- It is a property of the containing document object and contains references to all of the form objects in the document, in source order.
- We cannot add a form to the array, replace a form in the array, or remove a form from the array.

6. The frame Object:

- This object is a reflection of a frame element:

```
<frame>
```

The frames Array:

- A frames array contains the non-empty frames within a frame or a window.

7. The hidden Object:

- This object is a reflection of an input element with a type attribute of "hidden".

```
<input type="hidden">
```

8. The link Object:

- This object is a reflection of an a element that has an href attribute.

```
<a href=url>anchorText</a>
```

- All links are members of the document's links array in source order.

The links Array:

- It contains references to all the links in the document, in source order.
- We cannot remove a link from the array or replace a link in the array and we can create another links with the string object's link() method.

9. The password Object:

- This object is a reflection of an input element with a type attribute of "password".

```
<input type="password">
```

- The data entered by the user is not visible to the display each and every character appears as an asterisk (*) and it is not visible programmatically.
- Password objects are members of the containing form object's elements array.

10. The radio Object:

- This object represents a set of input elements of type "radio" with the same name attribute:

```
<input type="radio" name=radioName>
```

- Each button is a radio object is a member of the containing form object's elements array.

11. The reset Object:

- This object is a reflection of an input tag with a type attribute of "reset".

```
<input type="reset">
```

- Reset objects are members of the containing form object's elements array.

12. The select Object:

- This object is a reflection of a SELECT element of form.

```
<select><option>…</select>
```

- Select objects are members of the containing form object's elements array.

  The options Array:

  - This is a property of select objects and it allows we to manipulate the options of the select object. Individual options objects are reflections of option elements.

```
<option>, text to be displayed ……
```

13. The submit Object:

- This object is a reflection of an input element with a type attribute of "submit".

```
<input type="submit">
```

- Submit objects are members of the containing form object's elements array.

14. The text Object:

- This object is a reflection of an input element with a type attribute of "text".

```
<input type="text">
```

- Text object are members of the containing form object's elements array.

15. The textarea Object:

- This object is a reflection of textarea element like:

```
<textarea> text to be displayed…….</textarea>
```

3.8 EVENT HANDLING IN JAVASCRIPT [April 18]

- The change in the state of an object is known as an Event. In HTML, there are various events which represents that some activity is performed by the user or by the browser.
- When JavaScript code is included in HTML, js react over these events and allow the execution. This process of reacting over the events is called Event Handling.
- Thus, JS handles the HTML events via Event Handlers. For example, when a user clicks over the browser, add js code, which will execute the task to be performed on the event.

1. **Events:**

- JavaScript-enabled Web pages are typically event driven. Events are actions that occur on the webpage.
- Events are signals generated when specific action occurs.
- JavaScript is aware of these signals are scripts can be built to react to these events.

2. **Event Handler:**

- Event handlers execute JavaScript code to respond to events whenever they occur.
- Event handlers are scripts, in the form of attributes of specific HTML tags, which we as the programmer can write.
- The general **form/syntax** of an event handler is given below:

```
<html_tag other_attributes eventhandler = "JavaScript program">
```

- Event handler is actually a call to a function defined in the header of the document or a single JavaScript command. While any JavaScript statements, methods or functions can appear inside the quotation marks of event handler.

3. **Creating an Event Handler:**

- We do not need the <script> tag to define an event handler. Instead, we add an event attribute to an individual HTML tag.
- For example, here is a link that includes an OnMouseOver event handler.

```
<a href=http://www.pragati.com/ OnMouseOver="window.alert
                              ("We moved over the link");">Click here </a>
```

  Here, <a> is tag, which specifies a statement to be used as OnMouseOver event handler for the link and this statement displays an alert message when the mouse moves over the link.

- We can use JavaScript statements like the previous one in an event handler, but if we need more than one statement, its good idea to use a function as the event handler like this:

```
<a href = "#bottom" OnMouseOver = "DoIt();">
```

- Move the mouse over this link . This example calls a function called DoIt() when the user moves the mouse over the link.

Event Handlers with JavaScript: **[April 18]**

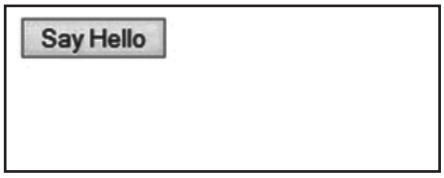
- Instead of specifying an event handler is an HTML document, we can use JavaScript to assign a function as an event handler. This allows us to set event handlers.
- To define event handler in this way, first define a function and then assign it is an event handler. Events are occurrences generated by the browser, such as loading a document or by the user such as moving the mouse.
- They are the user and browser activities to which we may respond dynamically with a scripting language like JavaScript.

1. **onclick Event:**
   - onclick is the most frequently used event type which occurs when a user clicks mouse left button.
   - We can put our validation, warning and so on against this event type.

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <script type="text/javascript">
         function sayHello()
         {
            alert("Hello JavaScript!")
         }
      </script>
   </head>
   <body>
      <input type="button" onclick="sayHello()" value="Say Hello" />
   </body>
</html>
```

Output:



After clicking Say Hello button we will get the following alert box:

2. **onsubmit Event:**

- This event occurs when we try to submit a form. So we can put our form validation against this event type.
- Following is a simple example showing its usage. Here, we are calling a validate() function before submitting a form data to the webserver. If validate() function returns true the form will be submitted otherwise it will not submit the data.

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <script type="text/javascript">
         <!--
         function validation()
         {
         all validation goes here
         .........
         return either true or false
         }
         //-->
      </script>
   </head>
   <body>
      <form method="POST" action="t.cgi" onsubmit="return validate()">
         .......
      <input type="submit" value="Submit" />
   </form>
   </body>
</html>
```

3. onmouseover and onmouseout Events:

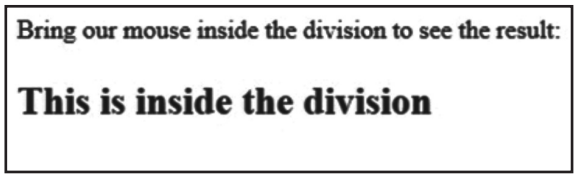
- These two event types will help us to create nice effects with images or even with text as well.
- The onMouseover event occurs when you bring mouse over any element and the onMouseout occurs when we take our mouse out from that element.

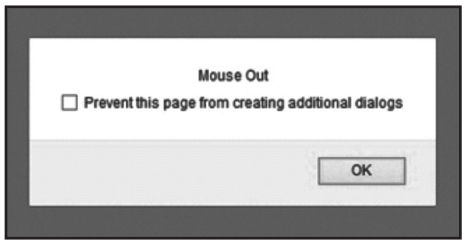
Example:

```
<!DOCTYPE html>
<html>
   <head>
      <script type="text/javascript">
         function over()
         {
            alert("Mouse Over");
         }
```

```
            function out()
            {
               alert("Mouse Out");
            }
         </script>
      </head>
      <body>
         <p>Bring our mouse inside the division to see the result:</p>
         <div onmouseover="over()" onmouseout="out()">
         <h2> This is inside the division </h2>
         </div>
      </body>
   </html>
```

Output:





4. onload and onunload Events:

- These two events are triggered when the user enters or leaves the page.
- The onload event can be used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.
- The onload and onunload events can be used to deal with cookies.

Example:

```
<!DOCTYPE html>
<html>
   <body onload="checkCookies()">
      <script>
         function checkCookies()
         {
```

```
            if(navigator.cookieEnabled==true)
            {
               alert("Cookies are enabled")
            }
            else
            {
               alert("Cookies are not enabled")
            }
            }
         </script>
            <p>An alert box should tell we if our browser
                                                has enabled cookies or not.</p>
      </body>
   </html>
```

Output:

- The above output will come if your browser has enabled cookies.

5. onchange Event:

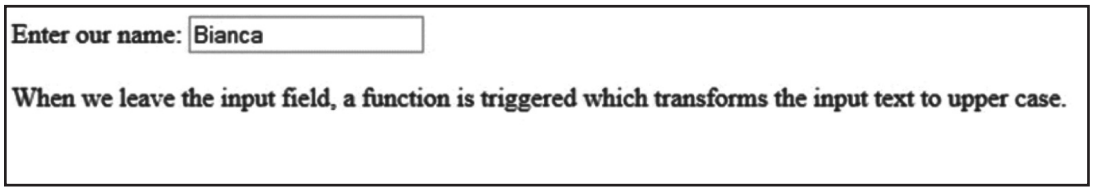
- This event are often used in combination with validation of input fields.
- Following is an example of how to use the onchange. The upperCase() function will be called when a user changes the content of an input field.

Example:

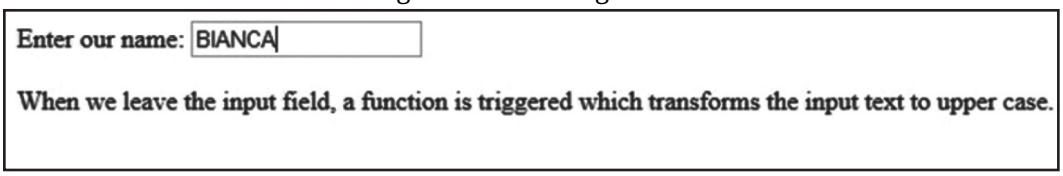
```
<!DOCTYPE html>
<html>
   <head>
      <script>
         function myFunction()
         {
            var x=document.getElementById("fname");
            x.value=x.value.toUpperCase();
         }
```

```
        </script>
    </head>
    <body>
        Enter our name: <input type="text" id="fname"
                                                onchange= "myFunction()">
        <p>When we leave the input field, a function is triggered
                        which transforms the input text to upper case.</p>
    </body>
</html>
```

Output:



- When we the text field we will get the following screen.



6. onmousedown, onmouseup and onclick Events:

- These three events are all parts of a mouse-click.
- First when a mouse-button is clicked, the onmousedown event is triggered, then, when the mouse-button is released, the onmouseup event is triggered, finally, when the mouse-click is completed, the onclick event is triggered.

Example:

```
<!DOCTYPE html>
<html>
    <head>
        <script>
            function mDown(obj)
            {
                obj.style.backgroundColor="#1ec5e5";
                obj.innerHTML="Release Me"
            }
```

```
            function mUp(obj)
            {
                obj.style.backgroundColor="#D94A38";
                obj.innerHTML="Thank You"
            }
        </script>
    </head>
    <body>
        <div onmousedown="mDown(this)" onmouseup="mUp(this"
                        style="background color:#D94A38;width:90px;
                                height: 20px;padding:40px;">Click Me</div>
    </body>
</html>
```

3.9 JS STRINGS

- The JavaScript string is an object that represents a sequence of characters. The string object is used for storing and manipulating text.
- A string can be any text inside quotes and we can use simple or double quotes as:

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

- The String object let's we work with a series of characters and wraps JavaScript's string primitive data type with a number of helper methods.
- Because JavaScript automatically converts between string primitives and String objects, we can call any of the helper methods of the String object on a string primitive.
- There are two ways to create string in JavaScript i.e., by string literal and By string object (using new keyword).

1. By String Literal:

- The string literal is created using double quotes. The **syntax of creating string using string literal** is given below:

```
var stringname="string value";
```

Example:

```
<script>
    var str=" Hello JavaScript ";
    document.write(str);
</script>
```

2. By String Object (Using new Keyword):

- The **syntax of creating string object using new keyword** is given below:

```
var stringname=new String("string literal");
```

Here, new keyword is used to create instance of string.

- The string parameter is series of characters that has been properly encoded.

Example:

```
<script>
   var stringname=new String("Hello JavaScript ");
   document.write(stringname);
</script>
```

String Properties:

Sr. No.	Property	Description
1.	`constructor`	This property returns a reference to the String function that created the object.
2.	`length`	This property returns the length of the string.
3.	`prototype`	This property allows you to add properties and methods to an object.

3.10 JS STRING METHODS

- Following table lists methods of string in JS:

Sr. No.	Method Name	Description
1.	`charAt()`	This method returns the character at the specified index.
2.	`charCodeAt()`	This method returns a number indicating the Unicode value of the character at the given index.
3.	`concat()`	This method combines the text of two strings and returns a new string.
4.	`indexOf()`	This method returns the index within the calling String object of the first occurrence of the specified value, or –1 if not found.
5.	`lastIndexOf()`	This method returns the index within the calling String object of the last occurrence of the specified value, or –1 if not found.

contd. ...

6.	`localeCompare()`	This method returns a number indicating whether a reference string comes before or after or is the same as the given string in sort order.
7.	`match()`	This method used to match a regular expression against a string.
8.	`replace()`	This method used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.
9.	`search()`	This method executes the search for a match between a regular expression and a specified string.
10.	`slice()`	This method extracts a section of a string and returns a new string.
11.	`split()`	This method Splits a String object into an array of strings by separating the string into substrings.
12.	`substr()`	This method returns the characters in a string beginning at the specified location through the specified number of characters.
13.	`substring()`	This method returns the characters in a string between two indexes into the string.
14.	`toLocaleLowerCase()`	This method returns the characters within a string are converted to lower case while respecting the current locale.
15.	`toLocaleUpperCase()`	This method returns the characters within a string are converted to upper case while respecting the current locale.
16.	`toLowerCase()`	This method returns the calling string value converted to lower case.
17.	`toString()`	This method returns a string representing the specified object.
18.	`toUpperCase()`	This method returns the calling string value converted to uppercase.
19.	`valueOf()`	This method returns the primitive value of the specified object.

PROGRAMS

Program 1: The length of a string (a String object) is found in the built in property length().

```
<!DOCTYPE html>
<html>
   <body>
      <script>
         var txt = "Hello JavaScript!";
         document.write("<p>" + txt.length + "</p>");
      </script>
   </body>
</html>
```

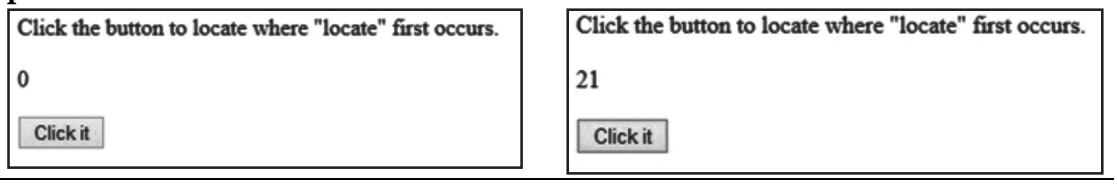
Output:

```
17
```

Program 2: The indexOf() method returns the position (as a number) of the first found occurrence of a specified text inside a string.

```
<!DOCTYPE html>
<html>
   <head>
      <script>
         function myFunction()
         {
            var str=document.getElementById("p1").innerHTML;
            var n=str.indexOf("locate");
            document.getElementById("p2").innerHTML=n+1;
         }
      </script>
   </head>
   <body>
      <p id="p1">Click the button to locate where "locate"
                                                        first occurs.</p>
      <p id="p2">0</p>
      <button onclick="myFunction()">Click it</button>
   </body>
</html>
```

Output:



Program 3: The match() method can be used to search for a matching content in a string.

```
<!DOCTYPE html>
<html>
   <body>
      <script>
         var str="Hello JavaScript world!";
         document.write(str.match("world") + "<br>");
         document.write(str.match("World") + "<br>");
         document.write(str.match("world!"));
      </script>
   </body>
</html>
```

Output:

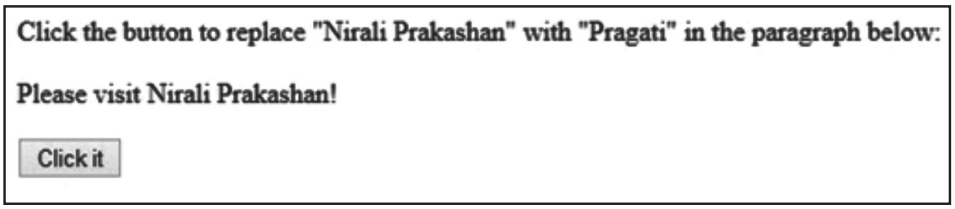
```
world
null
world!
```

Program 4: The replace() method replaces a specified value with another value in a string.

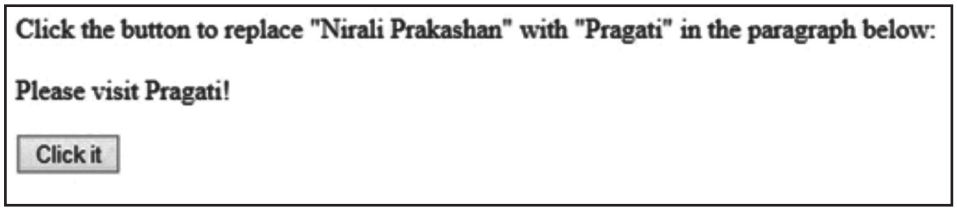
```
<!DOCTYPE html>
<html>
   <head>
      <script>
         function myFunction()
         {
            var str=document.getElementById("demo").innerHTML;
            var n=str.replace("Nirali Prakashan","Pragati");
            document.getElementById("demo").innerHTML=n;
         }
      </script>
   </head>
   <body>
```

```
        <p>Click the button to replace "Nirali Prakashan"
                                    with "Pragati" in the paragraph below:</p>
        <p id="demo">Please visit Nirali Prakashan!</p>
        <button onclick="myFunction()">Click it</button>
    </body>
</html>
```

Output:



After you click on the button you will get the following screen:



Program 5: A string is converted to upper/lower case with the methods toUpperCase() / toLowerCase().

```
<!DOCTYPE html>
<html>
    <body>
        <script>
            var txt="Hello JavaScript World!";
            document.write("<p>" + txt.toUpperCase() + "</p>");
            document.write("<p>" + txt.toLowerCase() + "</p>");
            document.write("<p>" + txt + "</p>");
        </script>
        <p>
            The methods returns a new string.
            The original string is not changed.
        </p>
    </body>
</html>
```

Output:

```
HELLO JAVASCRIPT WORLD!
hello javascript world!
Hello JavaScript World!
```

The methods returns a new string. The original string is not changed.

Program 6: Program for concatenating two string.

```
<html>
   <head>
      <title>JavaScript String concat() Method</title>
   </head>
   <body>
      <script type="text/javascript">
         var str1 = new String( "Amar" );
         var str2 = new String( "Salunkhe" );
         var str3 = str1.concat( str2 );
         document.write("Concatenated String:" + str3);
      </script>
   </body>
</html>
```

Output:

```
Concatenated String:AmarSalunkhe
```

Program 7: Program for charat().

```
<html>
   <head>
      <title>JavaScript String charAt() Method</title>
   </head>
   <body>
      <script type="text/javascript">
         var str = new String( "This is string" );
         document.writeln("str.charAt(0) is:" + str.charAt(0));
         document.writeln("<br />str.charAt(1) is:" + str.charAt(1));
         document.writeln("<br />str.charAt(2) is:" + str.charAt(2));
         document.writeln("<br />str.charAt(3) is:" + str.charAt(3));
         document.writeln("<br />str.charAt(4) is:" + str.charAt(4));
         document.writeln("<br />str.charAt(5) is:" + str.charAt(5));
      </script>
   </body>
</html>
```

Output:

```
str.charAt(0) is:T
str.charAt(1) is:h
str.charAt(2) is:i
str.charAt(3) is:s
str.charAt(4) is:
str.charAt(5) is:i
```

3.11 JS POP-UP BOXES (ALERT, CONFIRM AND PROMPT)

- When we want something highlighted, prompted or any important thing to say to users, so we can use Popup Boxes.
- In JavaScript we can create three kinds of popup boxes i.e., Alert Box, Confirm Box, and Prompt Box.

1. Alert Box: **[April 16, Oct. 18]**

- An alert box is often used if we want to make sure information comes through to the user.
- When an alert box pops up, the user will have to click "OK" to proceed.

Syntax: `alert("Alert Message")`

Example:

```
<html>
   <head>
      <script language="javascript">
         function show_alert()
         {
            alert("Hi! This is AlertBox!!")
         }
      </script>
   </head>
   <body>
      <input type="button" onclick="show alert()" value="Show alert box" >
   </body>
</html>
```

- In above example when user click on button the function show_alert() will be called, which display an alert box with the message “Hi! This is AlertBox!!” and one Ok button.

2. Confirm Box: **[April 18, Oct. 18]**

- A confirm box is often used if you want the user to verify or accept something.

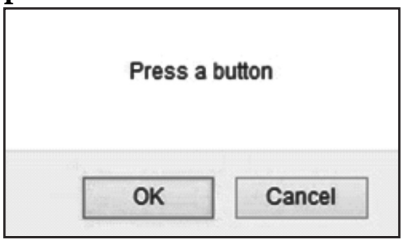
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax: `confirm("Message")`

Example:

```
<html>
   <head>
      <script language="javascript">
         function disp_confirm()
         {
            var r=confirm("Press a button")
         if (r==true)
         {
            document.write("You pressed OK!")
         }
         else
         {
            document.write("You pressed Cancel!")
         }
         }
      </script>
   </head>
   <body>
      <input type="button" onclick="disp_confirm()"
                                          value="Display a confirm box" />
   </body>
</html>
```

Output:



- In above example when user gets box with ok and cancel when user click on ok then text shown you pressed ok if cancel clicked then you pressed cancel.

3. Prompt Box: [Oct. 18]

- A prompt box is often used if you want the user to input a value before entering a page.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax: `prompt("sometext","defaultvalue")`

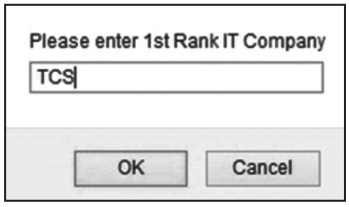
Example:

```
<html>
   <head>
      <script language="javascript">
         function disp_prompt()
         {
            var c=prompt("Please enter 1st Rank IT Company","Infosys")
            if (c!=null && c!="")
            {
               document.write("Hello, " + c + " is the best IT company.")
            }
         }
      </script>
   </head>
   <body>
      <input type="button" onclick="disp_prompt()"
                                          value="Display a prompt box">
   </body>
</html>
```

Output:

After clicking button this asked to enter value from user but also it's come with default value which is Visions Developer.

And after getting proper input from user print that value as given in function.



After we click OK we will get the following text:

```
Hello, TCS is the best IT company.
```

3.12 JQUERY LIBRARIES

- jQuery was first released in January 2006 by John Resig. jQuery is one of the most popular JavaScript libraries.
- jQuery is a JavaScript Library. JavaScript libraries contain various functions, methods, or objects to perform practical tasks on a webpage or JS-based application.
- jQuery is a lightweight, open-source JavaScript library that helps us build interactive web pages with animations, visual effects, and advanced functionality.
- jQuery is the most popular JavaScript library, used by around 70 million websites worldwide.
- The jQuery motto is “write less, do more”, because it reduces many lines of raw JavaScript code into a single line with its simple interface.
- The jQuery library is made up of selectors, event handlers and DOM traversal helpers. Along with Ajax, jQuery does just about everything we need JavaScript to do for the Web page.
- There are three most important elements that make jQuery work are explained below:
  1. **$()** or **jQuery():** The $() exists for the sole purpose of making it so you don’t have to write out jQuery() every single time you would like to use a selector.
  2. **selector:** This is how we select our DOM (Document Object Model) element. It’s the element we want to make changes to when the page loads.
  3. **action():** Tthis is the function that will tell the DOM what to do. It could be an event listener, it could be an effect depending on use case.
- Together, a simple basic jQuery **syntax** is written as:

  ```
  $(selector).action()
  ```

  - A $ sign to define/access jQuery.
  - A (selector) to "query (or find)" HTML elements.
  - A jQuery action() to be performed on the element(s).
- The jQuery syntax is tailor-made for selecting HTML elements and performing some action on the element(s).
- The principles of developing with jQuery are:
  1. **Separation of JavaScript and HTML:** The jQuery library provides simple syntax for adding event handlers to the DOM using JavaScript, rather than adding HTML event attributes to call JavaScript functions. Thus, it encourages developers to completely separate JavaScript code from HTML markup.
  2. **Brevity and Clarity:** jQuery promotes brevity and clarity with features like "chainable" functions and shorthand function names.

3. **Elimination of Cross-browser Incompatibilities:** The JavaScript engines of different browsers differ slightly so JavaScript code that works for one browser may not work for another. Like other JavaScript toolkits, jQuery handles all these cross-browser inconsistencies and provides a consistent interface that works across different browsers.
4. **Extensibility:** New events, elements, and methods can be easily added and then reused as a plug-in.

- The jQuery uses the Document Object Model (DOM) to manipulate, traverse, and select elements.
- The HTML document is loaded onto the DOM, where the browser creates a node tree when the page loads.
- A familial, hierarchical relationship is formed with the elements on the tree where elements are parents, children and siblings of each other.

3.13 INCLUDING JQUERY LIBRARY IN PAGE

- jQuery is a widely used JavaScript library. It simplifies client-side scripting of HTML.
- With the help of an easy-to-use API that works across a multitude of browsers, jQuery makes things like HTML document traversal and manipulation, event handling, animation and Ajax much simpler and easier.
- jQuery is embedded into the <script> tag of HTML file between the <head> tag and the <title> tag.
- There are following two ways to use jQuery:
  1. **Local Installation:** In this method, we have to download the jQuery js file and include it using a simple <script > tag within the HTML code. We can download jQuery library on the local machine and include it in the HTML code.
  2. **CDN Based Version:** Another method to include the jQuery to HTML is including the jQuery by CDN (Content Delivery Network). We can include jQuery library into the HTML code directly from CDN.

Local Installation of JQuery:

- Go to the https://jquery.com/download/ to download the latest version available. Now put downloaded jquery-2.1.3.min.js file in a directory of the website, e.g. /jquery. Then we can include jQuery library in the HTML file as follows :

```
<html>
   <head>
      <title>First jQuery Example</title>
          <script type = "text/javascript"
                                          src = "/jquery/jquery-2.1.3.min.js">
```

```
            </script>
            <script type = "text/javascript">
               $(document).ready(function()
               {
                  document.write("Hello, jQuery World!");
               });
            </script>
      </head>
      <body>
         <h1>Hello</h1>
      </body>
   </html>
```

Output:

```
Hello, jQuery World!
```

CDN Based Version of JQuery:

- We can include jQuery library into the HTML code directly from Content Delivery Network (CDN).
- Google and Microsoft provides content deliver for the latest version. Now let us rewrite above example using jQuery library from Google CDN.

```
<html>
   <head>
      <title>The jQuery Example</title>
          <script type = "text/javascript"
             src = "https://ajax.googleapis.com/ajax/libs/
                                                jquery/2.1.3/jquery.min.js">
          </script>
              <script type = "text/javascript">
                 $(document).ready(function()
                 {
                    document.write("Hello, jQuery World!");
                 });
              </script>
      </head>
      <body>
         <h1>Hello</h1>
      </body>
   </html>
```

Output:

```
Hello, jQuery World!
```

- We might have noticed that all jQuery methods in our examples, are inside a document ready event:

```
$(document).ready(function()
{
  // jQuery methods go here...
});
```

- This is to prevent any jQuery code from running before the document is finished loading (is ready).

3.14 JQUERY SELECTORS

- jQuery selectors are one of the most important parts of the jQuery library. jQuery Selectors are used to select and manipulate HTML elements.
- jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more.
- It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.
- A jQuery Selector is a function which makes use of expressions to find out matching elements from a DOM based on the given criteria.
- Simply we can say, selectors are used to select one or more HTML elements using jQuery. Once, an element is selected then we can perform various operations on that selected element.
- All jQuery selectors start with a dollor sign and parenthesis e.g. $(). It is known as the factory function.
- The factory function $() makes use of following three building blocks while selecting elements in a given document:
  1. **Tag Name** represents a tag name available in the DOM. For example $('p') selects all paragraphs <p> in the document.
  2. **Tag ID** represents a tag available with the given ID in the DOM. For example $('#some-id') selects the single element in the document that has an ID of some-id.
  3. **Tag Class** represents a tag available with the given class in the DOM. For example $('.some-class') selects all elements in the document that have a class of some-class.
- Following is an example which makes use of Tag Selector. This would select all the elements with a tag name p.

```
<html>
   <head>
      <title>jQuery Example</title>
          <script type = "text/javascript"
                        src ="https://ajax.googleapis.com/ajax/libs/
                                                    jquery/2.1.3/jquery.min.js">
```

```
            </script>
               <script type = "text/javascript"
                                          language = "javascript">
                  $(document).ready(function()
                  {
                     $("p").css("background-color", "red");
                  });
               </script>
            </head>
            <body>
               <div>
                   <p class = "myclass">This is a 1st paragraph...</p>
                   <p id = "myid">This is 2nd paragraph...</p>
                   <p>This is 3rd paragraph...</p>
               </div>
            </body>
         </html>
```

Output:

```
This is a 1st paragraph...

This is 2nd paragraph...

This is 3rd paragraph...
```

How to Use Selectors?

- The selectors in JQuery are very useful and would be required at every step while using jQuery. They get the exact element that we want from the HTML document.
- Few basic selectors in JQuery are explained below:
  - **Name selector** selects all elements which match with the given element Name.
  - **#ID selector** selects a single element which matches with the given ID.
  - **.Class selector** selects all elements which match with the given Class.
  - **Universal (*) selector** selects all elements available in a DOM.

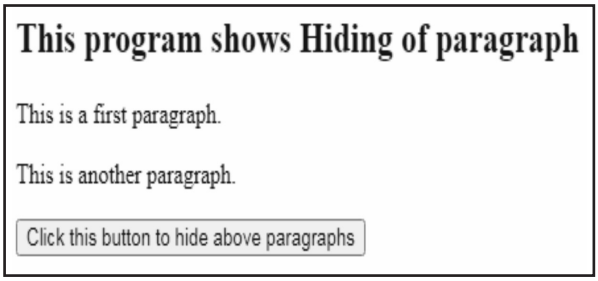
1. The element Selector in jQuery:

- The jQuery element selector selects elements based on the element name.
- The **syntax** is, `$('tagname')`
- Example, `$("p")`

- In following example, when a user clicks on a button, all <p> elements will be hidden:

```
<html>
   <head>
      <script src="https://ajax.googleapis.com/ajax/libs/
                                        jquery/3.5.1/jquery.min.js"></script>
      <script>
         $(document).ready(function(){
         $("button").click(function(){
         $("p").hide();
         });
         });
      </script>
   </head>
   <body>
      <h2>This program shows Hiding of paragraph</h2>
      <p>This is a first paragraph.</p>
      <p>This is another paragraph.</p>
      <button>Click this button to hide above paragraphs</button>
   </body>
</html>
```

Output:



After Clicking on Button:

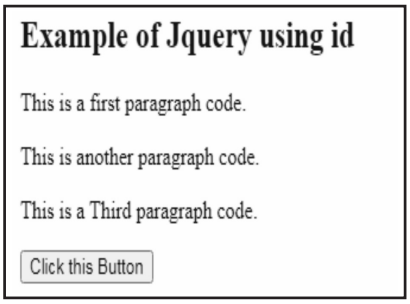
2. The #id Selector in jQuery:

- The jQuery #id selector uses the id attribute of an HTML tag to find the specific element.
- An id should be unique within a page, so we should use the #id selector when we want to find a single, unique element.

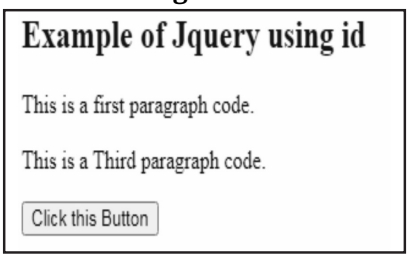
- The **syntax** is, `$('elementid').`
- Example, to find an element with a specific id, write a hash character, followed by the id of the HTML element such as, $("#test").
- In following example, when a user clicks on a button, the element with id="test" will be hidden:

```
<!DOCTYPE html>
<html>
   <head>
      <script src="https://ajax.googleapis.com/ajax/libs/
                                        jquery/3.5.1/jquery.min.js"></script>
      <script>
         $(document).ready(function(){
         $("button").click(function(){
         $("#test").hide();
         });
      });
      </script>
   </head>
   <body>
      <h2>Example of Jquery using id</h2>
      <p>This is a first paragraph code.</p>
      <p id="test">This is another paragraph code.</p>
      <p>This is a Third paragraph code.</p>
      <button>Click this Button</button>
   </body>
</html>
```

Output:



After Clicking on Button:

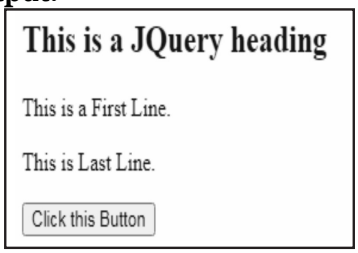


3. The .class Selector in jQuery:

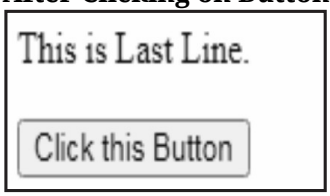
- The jQuery .class selector finds elements with a specific class.
- The **syntax** is, `$('.classid')`
- Example, to find an element with a specific class, write a period character, followed by the name of the class such as, $(".test")
- In following example, when a user clicks on a button, the elements with class="test" will be hidden:

```
<!DOCTYPE html>
<html>
   <head>
      <script src="https://ajax.googleapis.com/ajax/libs/
                                        jquery/3.5.1/jquery.min.js"></script>
      <script>
         $(document).ready(function(){
         $("button").click(function(){
         $(".tmp").hide();
         });
      });
      </script>
   </head>
   <body>
      <h2 class="tmp">This is a JQuery heading</h2>
      <p class="tmp">This is a First Line.</p>
      <p>This is Last Line.</p>
      <button>Click this Button</button>
   </body>
</html>
```

Output:



After Clicking on Button:



3.15 DOCUMENT OBJECT MODEL (DOM) MANIPULATION

- JQuery provides methods like such as .attr(), .html() etc., to manipulate DOM in efficient way.
- jQuery was designed to handle Browser Incompatibilities and to simplify HTML DOM Manipulation, Event Handling, Animations and Ajax.

Definition of DOM:

- "The Document Object Model (DOM) is an application programming interface (API) for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated." **OR**
- "The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."
- DOM defines the objects and properties and methods (interface) to access all XML elements. It is separated into following three different parts / levels:
  1. **Core DOM:** It is the standard model for any structured document.
  2. **XML DOM:** It is the standard model for XML documents.
  3. **HTML DOM:** It is the standard model for HTML documents.
- The Document Object Model (DOM) is a tree structure of various elements of HTML as follows:

```
<html>
   <head>
      <title>The jQuery Example</title>
   </head>
   <body>
      <div>
         <p>This is Line one .</p>
         <p>This is Line two.</p>
         <p>This is Line three.</p>
      </div>
   </body>
</html>
```

Output:

```
This is Line one .
This is Line two.
This is Line three.
```

- Following are the important points about the above program structure:
  - The <html> is the ancestor of all the other elements; in other words, all the other elements are descendants of <html>.
  - The <head> and <body> elements are not only descendants, but children of <html>, as well.
  - Likewise, in addition to being the ancestor of <head> and <body>, <html> is also their parent.
  - The <p> elements are children (and descendants) of , descendants of <body> and <html>, and siblings of each other <p> elements.
- One very important part of jQuery is the possibility to manipulate the DOM. jQuery provides various methods to add, edit or delete DOM element(s) in the HTML page.
- jQuery contains powerful methods for changing and manipulating HTML elements and attributes.
- jQuery comes with a bunch of DOM related methods that make it easy to access and manipulate elements and attributes.
- jQuery provides methods to manipulate DOM in efficient way. We do not need to write big code to modify the value of any element's attribute or to extract HTML code from a paragraph or division.
- jQuery provides methods such as .attr(), .html() and .val() which act as getters, retrieving information from DOM elements for later use.
- jQuery attr() method is used to get or set the value of specified attribute of DOM element. jQuery html() method sets or returns the content of selected elements (including HTML markup).
- jQuery val() method sets or returns the value of form fields. jQuery text() method sets or returns the text content of selected elements.

Adding New HTML Content:

- We will look at following jQuery methods that are used to add new content:
  1. **append()** method inserts content at the end of the selected elements.
  2. **prepend()** method inserts content at the beginning of the selected elements.
  3. **after()** method inserts content after the selected elements.
  4. **before()** method inserts content before the selected elements.

Removing Elements/Content:

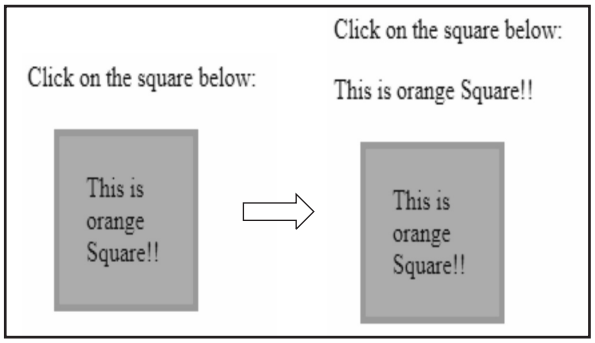
- With jQuery, it is easy to remove existing HTML elements. To remove elements and content, there are following two jQuery methods:
  1. **remove()** method removes the selected element (and its child elements).
  2. **empty()** method removes the child elements from the selected element.

Content Manipulation:

- In jQuery the .html() method gets the html contents (innerHTML) of the first matched element. The **syntax** for the method is, `selector.html()`.
- Following example, shows use of .html() and .text(val) methods. The .html() retrieves HTML content from the object and then .text(val) method sets value of the object using passed parameter.

```
<html>
   <head>
      <title>The jQuery Example for Content Manipulation</title>
          <script type = "text/javascript"
             src = "https://ajax.googleapis.com/ajax/libs/
                                                jquery/2.1.3/jquery.min.js">
          </script>
             <script type = "text/javascript" language = "javascript">
                $(document).ready(function()
                {
                   $("div").click(function ()
                   {
                      var content = $(this).html();
                      $("#result").text( content );
                   });
                });
          </script>
          <style>
             #division{ margin:20px;padding:20px; border:4px
                                                    solid #999; width:60px;}
          </style>
       </head>
       <body>
          <p>Click on the square below:</p>
          <span id = "result"> </span>
          <div id = "division" style = "background-color:orange;">
             This is orange Square!!
          </div>
       </body>
    </html>
```

Output:



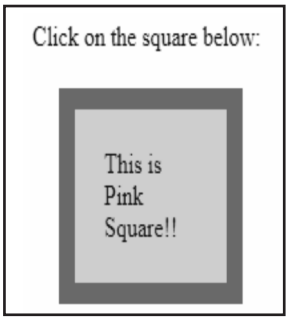
DOM Element Replacement:

- We can replace a complete DOM element with the specified HTML or DOM elements. The replaceWith(content) method serves this purpose very well.
- The **syntax** for this method is, `selector.replaceWith(content)`. Here, content is what we want to have instead of original element. This could be HTML or simple text.
- Following example would replace division element with "<h1>JQuery is </h1>":

```
<html>
   <head>
      <title>The jQuery Example for replace division element </title>
      <script type = "text/javascript"
         src ="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
      </script>
      <script type = "text/javascript" language = "javascript">
         $(document).ready(function() {
            $("div").click(function () {
               $(this).replaceWith("<h1>PHP using JQuery is Easy</h1>");
            });
         });
      </script>
      <style>
         #division{ margin:20px;padding:22px; border:12px solid #666;
                                                          width:70px;}
      </style>
   </head>
   <body>
      <p>Click on the square below:</p>
      <span id = "result"> </span>
```

```
        <div id = "division" style = "background-color:pink;">
           This is Pink Square!!
        </div>
     </body>
  </html>
```

Output:



After Click on square:

PHP using JQuery is Easy

Removing DOM Elements:

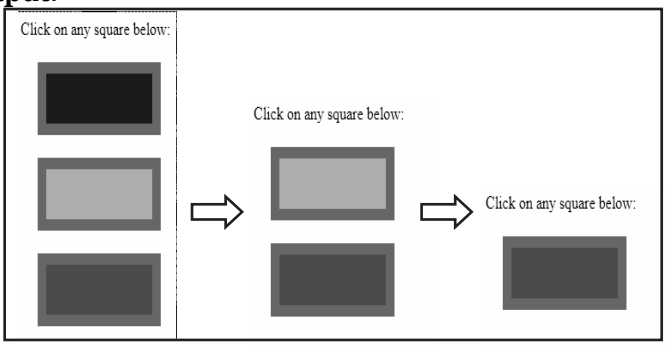
- There may be a situation/condition when we would like to remove one or more DOM elements from the document. jQuery provides two methods to handle the situation namely, empty() and remove(expr).
- The empty() method remove all child nodes from the set of matched elements where as the method remove(expr) method removes all matched elements from the DOM.
- The **syntax** for this method is, `selector.remove([expr])` OR `selector.empty()`. We can pass optional parameter expr to filter the set of elements to be removed.
- Following is an example of JQuery where elements are being removed as soon as they are clicked:

```
<html>
   <head>
      <title>The jQuery Example for removing Element</title>
      <script type = "text/javascript"
                     src = "https://ajax.googleapis.com/ajax/libs/
                                                jquery/2.1.3/jquery.min.js">
      </script>
      <script type = "text/javascript" language = "javascript">
         $(document).ready(function() {
            $("div").click(function () {
```

```
                $(this).remove( );
            });
        });
    </script>

    <style>
        .div{ margin:20px;padding:22px; border:10px solid #666;
                                                    width:80px;}
    </style>
  </head>
  <body>
    <p>Click on any square below:</p>
    <span id = "result"> </span>
    <div class = "div" style = "background-color:Blue;"></div>
    <div class = "div" style = "background-color:orange;"></div>
    <div class = "div" style = "background-color:green;"></div>
  </body>
</html>
```

Output:



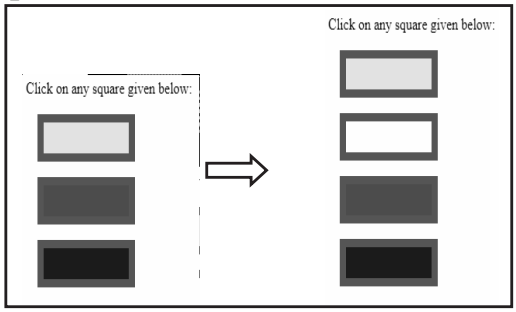
Inserting DOM Elements:

- There may be a situation/condition when we would like to insert new one or more DOM elements in the existing document. The jQuery provides various methods like after(content), before(content) etc. to insert elements at various locations.
- The after(content) method insert content after each of the matched elements where as the method before(content) method inserts content before each of the matched elements.
- The **syntax** for these methods are, `selector.after(content)` and `selector.before(content)`. Here, content is what you want to insert. This could be HTML or simple text.

- Following is an example where elements are being inserted just before the clicked element:

```
<html>
   <head>
      <title>The jQuery Example for <div> elements are being inserted
      </title>
      <script type = "text/javascript"
                     src = "https://ajax.googleapis.com/ajax/libs/
                                                jquery/2.1.3/jquery.min.js">
      </script>
      <script type = "text/javascript" language = "javascript">
         $(document).ready(function() {
            $("div").click(function () {
               $(this).before('<div class="div"></div>' );
            });
         });
      </script>
      <style>
         .div{ margin:15px;padding:15px; border:8px solid #555;
width:80px;}
      </style>
   </head>
   <body>
      <p>Click on any square given below:</p>
      <span id = "result"> </span>
      <div class = "div" style = "background-color:yellow;"></div>
      <div class = "div" style = "background-color:red;"></div>
      <div class = "div" style = "background-color:blue;"></div>
   </body>
</html>
```

Output:



DOM Manipulation Methods:

- Following are the methods which we can use to manipulate DOM elements:
  1. The `after(content)` method insert content after each of the matched elements.
  2. The `append(content)` method append content to the inside of every matched element.
  3. The `appendTo(selector)` method append all of the matched elements to another, specified, set of elements.
  4. The `before(content)` method inserts content before each of the matched elements.
  5. The `clone(bool)` method matched DOM elements and all their event handlers, and select the clones.
  6. The `clone()` method matched DOM elements and select the clones.
  7. The `empty()` method removes all child nodes from the set of matched elements.
  8. The `html(val)` method sets the html contents of every matched element.
  9. The `html()` method gets the html contents (innerHTML) of the first matched element.
  10. The `insertAfter(selector)` method inserts all of the matched elements after another, specified, set of elements.
  11. The `insertBefore(selector)` method inserts all of the matched elements before another, specified, set of elements.
  12. The `prepend(content)` method prepend content to the inside of every matched element.
  13. The `prependTo(selector)` method prepend all of the matched elements to another, specified, set of elements.
  14. The `remove(expr)` method removes all matched elements from the DOM.
  15. The `replaceAll(selector)` method replaces the elements matched by the specified selector with the matched elements.
  16. The `replaceWith(content)` method replaces all matched elements with the specified HTML or DOM elements.
  17. The `text(val)` method sets the text contents of all matched elements.
  18. The `text()` method gets the combined text contents of all matched elements.
  19. The `wrap(elem)` method wraps each matched element with the specified element.
  20. The `wrap(html)` method wraps each matched element with the specified HTML content.
  21. The `wrapAll(elem)` method wraps all the elements in the matched set into a single wrapper element.
  22. The `wrapAll(html)` method wraps all the elements in the matched set into a single wrapper element.

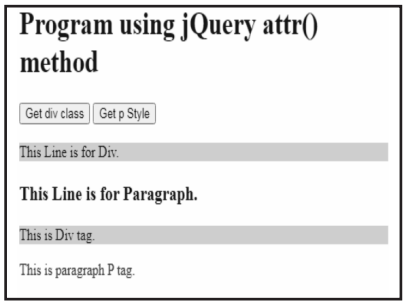
23. The `wrapInner(elem)` method wraps the inner child contents of each matched element (including text nodes) with a DOM element.
24. The `wrapInner(html)` method wraps the inner child contents of each matched element (including text nodes) with an HTML structure.
25. The `attr()` method is used to get attribute values.

Example on attr() Method of jQuery:

```
<!DOCTYPE html>
<html>
   <head>
      <script src="https://ajax.googleapis.com/ajax/libs/
                                          jquery/1.11.2/jquery.min.js">
      </script>
      <script>
         $(document).ready(function () {
            $('#btn_DivStyle').click(function(){
               alert($('div').attr('class'));
         });
         $('#btn_PStyle').click(function(){
            alert($('p').attr('style'));
         });
         $('div').attr('class','yellowDiv');
                              // adds class='yellowDiv' to each div element
      });
   </script>
   <style>
      .yellowDiv{
         background-color:pink;
      }
   </style>
   </head>
   <body>
      <h1>Program using jQuery attr() method</h1>
      <button id="btn_DivStyle">Get div class</button>
      <button id="btn_PStyle">Get p Style</button> </br></br>
      <div>
         This Line is for Div.
```

```
        </div>
        <p style="font-size:20px;font-weight:bold">
           This Line is for Paragraph.
        </p>
        <div>
           This is Div tag.
        </div>
        <p style="color:Blue">
           This is paragraph P tag.
        </p>
     </body>
  </html>
```

Output:

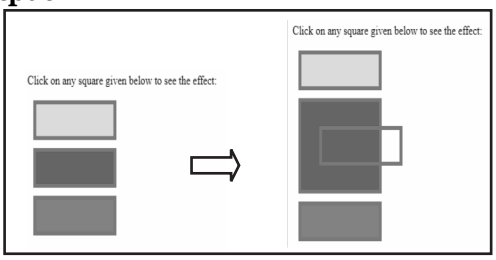


Example on append() Method of jQuery:

```
<html>
   <head>
      <title>The jQuery Example for Append method</title>
      <script type = "text/javascript"
        src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/
                                                        jquery.min.js">
      </script>
      <script type = "text/javascript" language = "javascript">
         $(document).ready(function() {
            $("div").click(function () {
               $(this).append('<div class = "div1"></div>' );
            });
         });
```

```
        </script>
        <style>
           .div1{margin:10px;padding:20px; border:5px solid #777;
                                                          width:90px;}
        </style>
     </head>
     <body>
        <p>Click on any square given below to see the effect:</p>
        <div class = "div1" style = "background-color:yellow;"></div>
        <div class = "div1" style = "background-color:red;"></div>
        <div class = "div1" style = "background-color:grey;"></div>
     </body>
  </html>
```

Output:



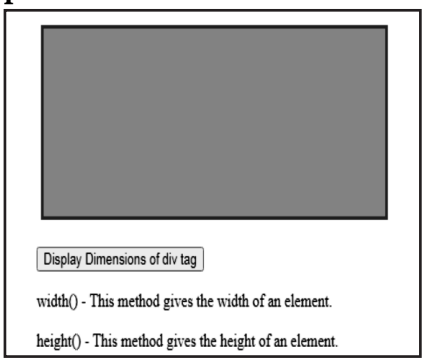
DOM Manipulation Methods in jQuery:

- jQuery provides various methods to add, edit or delete DOM element(s) in the HTML page.
- The jQuery library includes various methods to manipulate DOM element's dimensions like height, width, offset, position etc.
  1. The `width()` method sets or returns the width of an element (excludes padding, border and margin).
  2. The `height()` method sets or returns the height of an element (excludes padding, border and margin).
  3. The `innerWidth()` method returns the width of an element (includes padding).
  4. The `innerHeight()` method returns the height of an element (includes padding).
  5. The `outerWidth()` method returns the width of an element (includes padding and border).
  6. The `outerHeight()` method returns the height of an element (includes padding and border).
  7. The `offset()` method gets or sets left and top coordinates of the specified element(s).
  8. The `position()` method gets the current coordinates of the specified element(s).

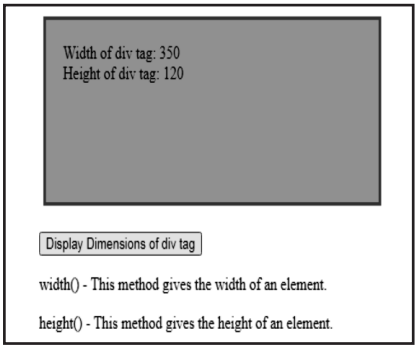
Example:

```
<!DOCTYPE html>
<html>
   <head>
      <script src="https://ajax.googleapis.com/ajax/libs/
                                    jquery/3.5.1/jquery.min.js"></script>
      <script>
         $(document).ready(function(){
            $("button").click(function(){
            var tmp = "";
            tmp += "Width of div tag: " + $("#div1").width() + "</br>";
            tmp += "Height of div tag: " + $("#div1").height();
         $("#div1").html(tmp);
         });
      });
   </script>
   <style>
      #div1 {
         height: 120px;
         width: 350px;
      padding: 20px;
      margin: 5px;
         background-color: orange;
         border: 3px solid green;
      }
   </style>
   </head>
      <body>
         <div id="div1"></div>
         <br>
         <button>Display Dimensions of div tag</button>
         <p>width() - This method gives the width of an element.</p>
         <p>height() - This method gives the height of an element.</p>
      </body>
</html>
```

Output:



After Clicking on Button:



CSS Manipulation using jQuery:

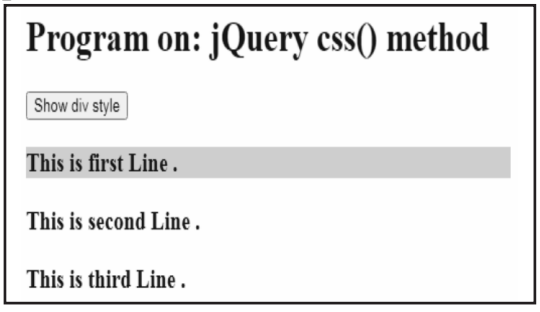
- The jQuery library includes various methods to manipulate style properties and CSS class of DOM element(s).
- jQuery methods for styling and CSS manipulation are given below:
  1. The `css()` method gets or set style properties to the specified element(s).
  2. The `addClass()` method adds one or more class to the specified element(s).
  3. The `hasClass()` method determines whether any of the specified elements are assigned the given CSS class.
  4. The `removeClass()` method removes a single class, multiple classes, or all classes from the specified element(s).
  5. The `toggleClass()` method toggles between adding/removing classes to the specified elements.

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <script src="https://ajax.googleapis.com/ajax/libs/
                                          jquery/1.11.2/jquery.min.js">
```

```
        </script>
            <script>
                $(document).ready(function () {
                $('#my_Div').css('background-color','pink');
                    $('#bt_Show').click(function(){
                    $('p').css({'font-size':'20px','font-weight':'bold'});
                    alert($('#my_Div').css('background-color'));
                });
          });
      </script>
  </head>
  <body>
     <h1>Program on: jQuery css() method</h1>
     <button id="bt_Show">Show div style</button>
     <div id="my_Div">
        <p>
           This is first Line.
        </p>
     </div>
     <div>
        <p>This is second Line .</p>
     </div>
        <div >
           <p>This is third Line .</p>
        </div>
     </body>
  </html>
```

Output:



Traversing DOM Elements using jQuery:

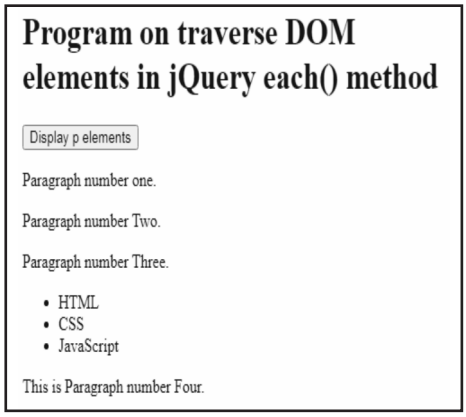
- The jQuery library includes various methods to traverse DOM elements in a DOM hierarchy.
- jQuery methods for traversing DOM elements are given below:
  1. The `children()` method gets all the child elements of the specified element(s).
  2. The `each()` method iterates over specified elements and execute specified call back function for each element.
  3. The `find()` gets all the specified child elements of each specified element(s).
  4. The `first()` method gets the first occurrence of the specified element.
  5. The `next()` method gets the immediately following sibling of the specified element.
  6. The `parent()` method gets the parent of the specified element(s).
  7. The `prev()` method gets the immediately preceding sibling of the specified element.
  8. The `siblings()` method gets the siblings of each specified element(s).

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <script src="https://ajax.googleapis.com/ajax/libs/
                                          jquery/1.11.2/jquery.min.js">
    </script>
    <script>
      $(document).ready(function () {
      $('#bnt_Show').click(function(){
            $('p').each(function (index) {
               alert('index: ' + index + ', text: ' + $(this).text());
            });
         });
       });
    </script>
</head>
<body>
   <h1>Program on traverse DOM elements in  jQuery each() method</h1>
   <button id="bnt_Show">Display p elements</button>
   <div>
      <p> Paragraph number one.</p>
```

```
        </div>
        <div id="my_Div">
            <p>Paragraph number Two.</p>
            <div id="inrDiv">
                <p>Paragraph number Three.</p>
            </div>
            <div>
                <ul>
                    <li>HTML</li>
                    <li>CSS</li>
                    <li>JavaScript</li>
                </ul>
            </div>
        </div>
            <div>
                <p>This is Paragraph number Four.</p>
            </div>
        </body>
    </html>
```

Output:



How to call a jQuery Library Functions?

- As almost everything, we do when using jQuery reads or manipulates the DOM, we need to make sure that we start adding events etc. as soon as the DOM is ready.
- If we want an event to work on your page, you should call it inside the $(document).ready() function.

- Everything inside it will load as soon as the DOM is loaded and before the page contents are loaded.
- To do this, we register a ready event for the document as given below:

```
$(document).ready(function()
{
   // do stuff when DOM is ready
});
```

- To call upon any jQuery library function, use HTML script tags as follows:

```
<html>
   <head>
      <title>The jQuery library function call Example</title>
      <script type = "text/javascript"
         src = "https://ajax.googleapis.com/ajax/libs/jquery/
                                                    2.1.3/jquery.min.js">
      </script>
      <script language = "javascript" type = "text/javascript" >
         $(document).ready(function() {
            $("div").click(function() {alert("Hello, world!");});
         });
      </script>
   </head>
   <body>
      <div id = "my_div">
         Click on this to see a dialogue box.
      </div>
   </body>
</html>
```

Output:

```
Click on this to see a dialogue box.
```

ADDITIONAL PROGRAMS

Program 1: Program to find average of three numbers.

```
<html>
<head>
   <title>Average of Three Numbers</title>
      <script language="JavaScript">
         function calcAvg()
         {
         var inpNum1 = 14;
         var inpNum2 = 28;
         var inpNum3 = 34;
         numAvg = doCalcAvg(inpNum1, inpNum2, inpNum3);
         alert("The average is: " + numAvg);
         }
            function doCalcAvg(inpNum1, inpNum2, inpNum3)
            {
               var ans;
               ans = (Number(inpNum1)+Number(inpNum2)+Number(inpNum3)/4;
               return (ans);
            }
      </script>
   </head>
   <body onLoad="calcAvg();">
   </body>
</html>
```

Output:

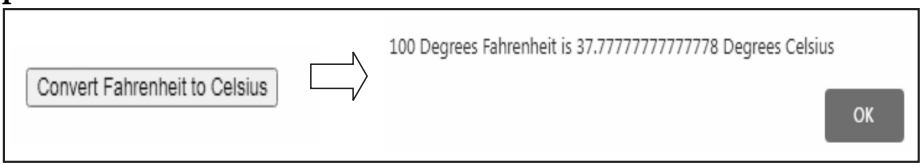
```
The average is: 25.33
```

Program 2: Program to convert Fahrenheit to Celsius.

```
<html>
   <head>
      <title>Conversion to Fahrenheit to Celsius </title>
         <script language="JavaScript">
            function inputFah()
            {
               var fah = 100;
```

```
                ansCel = doCelCalc(fah);
                alert(fah + " Degrees Fahrenheit is " + ansCel +
                                                    " Degrees Celsius");
            }
                function doCelCalc(fah)
                {
                    var ans = ((Number(fah) - 32) / 1.8);
                    return (ans);
                }
        </script>
    </head>
    <body>
        <input type="button" value="Convert Fahrenheit to Celsius"
                                                    onClick="inputFah();">
    </body>
</html>
```

Output:

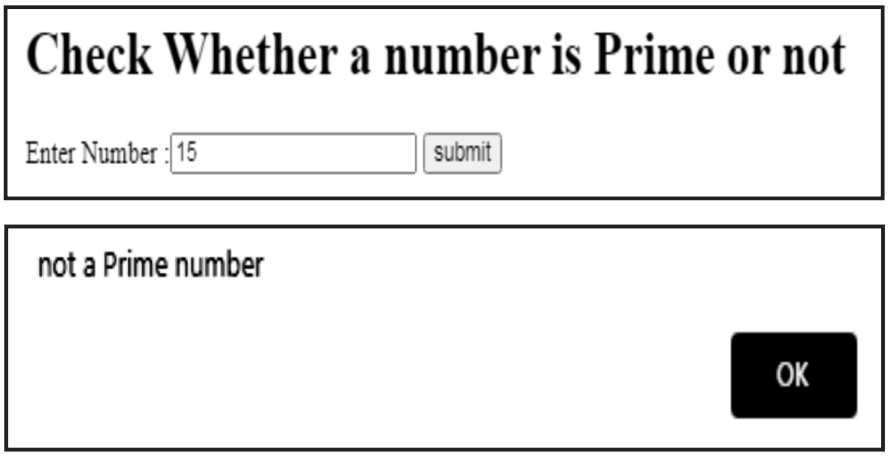


Program 3: Program to the given no is prime or not.

```
<html>
    <head>
        <script language="JavaScript">
            function Prime()
            {
               var i,flag=0,number;
               number = Number(document.getElementById("N").value);
            for(i=2; i <= number/2; i++)
            {
               if(number%i == 0)
               {
                  flag = 1;
                  break;
               }
               }
```

```
            if(flag == 0)
            {
               window.alert("prime number");
            }
            else
            {
                window.alert("not a Prime number");
            }
            }
         </script>
      </head>
      <body>
         <br>
         <h1>Check Whether a number is Prime or not</h1>
            Enter Number :<input type="text" name="n" id = "N"/>
         <button onClick="Prime()">submit</button>
      </body>
   </html>
```

Output:

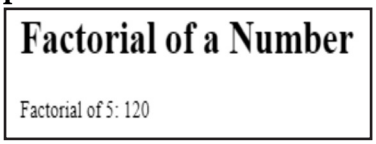


Program 4: Program to find factorial of the number.

```
<html>
   <body>
      <h1>Factorial of a Number</h1>
      <script language="JavaScript">
         var num = 5;
```

```
            var i = 1;
            var factorial = 1;
            for(i = 1; i <= num; i++)
            factorial *= i;
            if(num)
            document.write("Factorial of "+num+": "+factorial);
            else
            document.write("Factorial of 0: 0");
         </script>
      </body>
   </html>
```

Output:



Program 5: Program to find the year is leap or not.

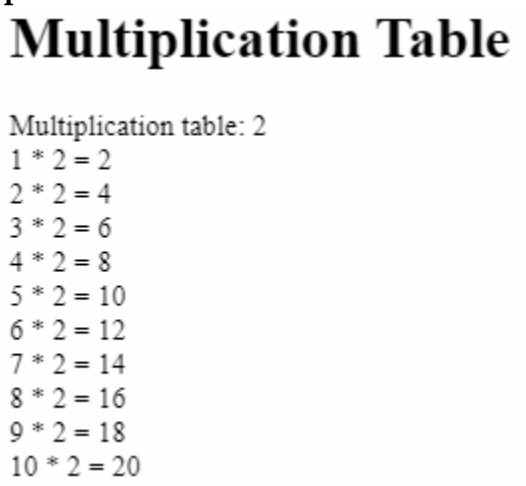
```
<html>
   <body>
      <h1>Leap Year or Not</h1>
      <script language="JavaScript">
         var year = 2012;
         if(year % 4 == 0)
         {
            if((year % 100 == 0) && (year % 400 != 0))
            document.write(year +" is not a leap year");
            else
            document.write(year +" is a leap year");
         }
         else
                document.write(year+" is not a leap year");
      </script>
   </body>
</html>
```

Output:

Program 6: Program for multiplication of table.

```
<html>
   <body>
      <h1>Multiplication Table</h1>
      <script language="JavaScript">
         var table = 9;
         var length = 10;
         var i = 1;
         document.write("Multiplication table: "+ table);
         for(i = 1; i <= length; i++)
         document.write("<br>"+i+" * "+table+" = " +(i * table));
      </script>
   </body>
</html>
```

Output:

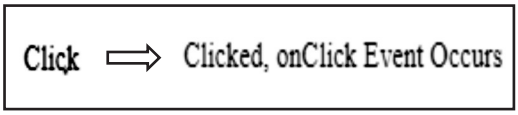


Program 7: Program for events.

```
<html>
   <body>
      <p onclick="myFunction()">Click</p>
         <script language="JavaScript">
            function myFunction()
            {
               document.write("Clicked, onClick Event ");
            }
```

```
            </script>
        </body>
    </html>
```

Output:



Program 8: Program for combining two sting.

```
<html>
   <body>
      <script language="JavaScript">
         var str1 = "Hello";
         var str2 = " ";
         var str3 = "Amar…";
         var str =str1.concat(str2, str3);
         document.write(str);
      </script>
   </body>
</html>
```

Output:

```
HelloAmar…
```

Program 9: Program for toString().

```
<html>
   <body>
      <script language="JavaScript">
         var str1 ="One";
         var str2 =new String("Two");
         document.write("str1 : " +typeof str1);
         document.write("<br>str2 : " +typeof str2);
      </script>
   </body>
</html>
```

Output:

```
str1 : string
str2 : object
```

Program 10: Program for conditional operator.

```
<html>
   <body>
      <script language="JavaScript">
         var mark = 35;
         var result = (mark < 50) ? "Fail":"Pass";
         document.write(result);
      </script>
   </body>
</html>
```

Output:

```
Fail
```

Program 11: Program for variables.

```
<html>
   <body>
      <script language="JavaScript">
         var x = "Amar";
         var y = "Akash";
         var z =  x + y;
         document.write("z = "+z);
      </script>
   </body>
</html>
```

Output:

```
z = AmarAkash
```

Program 12: Program to print following pattern:

```
 1
 1 2
 1 2 3
<html>
   <body>
      <script language="JavaScript">
         var i, j;
         for(i=1; i<=3; i++)
         {
           for(j=1; j<=i; j++)
```

```
                document.write(j);
                document.write("<br>");
            }
        </script>
    </body>
</html>
```

Output:

```
1
12
123
```

Program 13: Program for print following pattern:

```
  1
  0 1
  1 0 1
<html>
    <body>
        <script language="JavaScript">
            var i, j;
            for(i=1; i<=3; i++)
            {
               for(j=i; j>=1; j--)
                  document.write(j % 2);
                  document.write("<br>");
            }
        </script>
    </body>
</html>
```

Output:

```
1
01
101
```

Program 14: Program to print following pattern:

```
  1
  2 2
  3 3 3
<html>
```

```
    <body>
       <script language="JavaScript">
          var i, j;
          for(i=1; i<=3; i++)
          {
             for(j=1; j<=i; j++)
                document.write(i);
                document.write("<br>");
          }
       </script>
    </body>
</html>
```

Output:

```
1
22
333
```

Program 15: Program to print following pattern:

```
1 2 3 4 5
1 2 3
1
<html>
    <body>
       <script language="JavaScript">
          var i, j;
          for(i=5; i>=1; i-=2)
          {
             for(j=1; j<=i; j++)
                document.write(j);
                document.write("<br>");
          }
       </script>
    </body>
</html>
```

Output:

```
12345
123
1
```

Program 16: Program to print following pattern:

```
*
* *
* * *
```

```
<html>
   <body>
      <script language="JavaScript">
         var i, j;
         for(i=1; i<=3; i++)
         {
            for(j=1; j<=i; j++)
               document.write("*");
               document.write("<br>");
         }
      </script>
   </body>
</html>
```

Output:

```
*
**
***
```

Program 17: Program to print following pattern:

```
* * *
 * *
  *
```

```
<html>
   <body>
      <script language="JavaScript">
         var i, j, k;
         for(i=1; i<=5; i++)
         {
            for(j=1; j<i; j++)
               document.write("  ");
                for(k=3; k>=i; k--)
                   document.write("*");
                   document.write("<br>");
         }
```

```
        </script>
      </body>
   </html>
```

Output:

```
***
 **
  *
```

Program 18: Program to print following pattern:

```
    *
   ***
  *****
```

```
<html>
   <body>
      <script language="JavaScript">
         var i, j, k;
         for(i=1; i<=5; i++)
         {
            for(j=i; j<5; j++)
               document.write("  ");
                for(k=1; k<i*2; k++)
                   document.write("*");
                   document.write("<br>");
         }
      </script>
   </body>
</html>
```

Output:

```
    *
   ***
  *****
```

Program 19: Program #id selector in jQuery.

```
<html>
   <head>
      <script
   src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
      </script>
```

```
    </head>
    <body>
        <h4>jQuery Selector #id</h4>
        <p id="point">This is paragraph</p>
        <div>This is div</div>
            <script>
                $(document).ready(function()
                {
                    $("#point").css("color","green");
                });
            </script>
    </body>
</html>
```

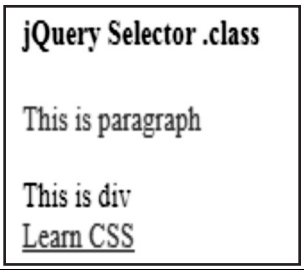
Output:

```
jQuery Selector #id
This is paragraph
This is div
```

Program 20: Program .class selector in jQuery.

```
<html>
    <head>
        <script
    src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
        </script>
    </head>
    <body>
        <h4>jQuery Selector .class</h4>
        <p class="point">This is paragraph</p>
        <div>This is div</div>
        <a href="/css" class="point">Learn CSS</a>
            <script>
                $(document).ready(function()
                {
                    $(".point").css("color","green");
                });
            </script>
    </body>
</html>
```

Output:



PRACTICE QUESTIONS

Q. I Multiple Choice Questions:

1. Which is a dynamic, lightweight, interpreted programming language?
   (a) JavaScript (b) JQuery
   (c) HTML (d) PHP
2. Advantages of JavaScript includes,
   (a) Less server interaction (b) Increased interactivity
   (c) Richer interfaces (d) All of the mentioned
3. The file extension of JavaScript is file is,
   (a) .jsp (b) .js
   (c) .jshtm (d) .jsph
4. Which is a JavaScript Library?
   (a) jScript (b) JpHP
   (c) JavaScript (d) jQuery
5. Which of the following jQuery method is used to attach a handler to an event?
   (a) bind() (b) bind()
   (c) attach() (d) None of the mentioned
6. The variables, in JavaScript declared with the,
   (a) var keyword (b) function keyword
   (c) const keyword (d) return keyword
7. Which statement is used to execute the code whether condition is true or false?
   (a) switch (b) for
   (c) break (d) if-else
8. Which statement in JavaScript is used to jumps out of a loop?
   (a) continue (b) exit
   (c) break (d) for
9. Which loop is used to loop through an object's properties?
   (a) for (b) do...while
   (c) while (d) for...in

10. In JavaScript which boxes prevent the user from accessing other aspects of a program until the popup is closed, so they should not be overused?
    (a) Popup (b) jsBoxex
    (c) ErrrorBoxes (d) None of the mentioned
11. Which are used to select one or more HTML elements using jQuery.
    (a) loops (b) statements
    (c) selectors (d) None of the mentioned
12. Which method in JavaScript is used to combines the text of two strings and returns a new string?
    (a) concat() (b) combine()
    (c) join() (d) concat()
13. Features supported by jQuery include,
    (a) AJAX Support (b) Cross Browser Support
    (c) Lightweight (d) All of the mentioned
14. All selectors in jQuery start with the,
    (a) @() (b) $()
    (c) ?() (d) %()
15. Which dialog box is mostly used to inform or alert the user by displaying some messages in a small dialogue box?
    (a) prompt (b) confirm
    (c) alart (d) All of the mentioned
16. A JavaScript function is defined with which keyword, followed by a name, followed by parentheses ().
    (a) function (b) var
    (c) function_create (d) fun_start
17. Which selector in jQuery used the id attribute of an HTML tag to find the specific element?
    (a) .class (b) this
    (c) * (d) #id
18. JavaScript's interaction with HTML is handled through _____, (that occur when the user or the browser manipulates an action such as mouseClick, keyPress).
    (a) functions (b) strings
    (c) events (d) None of the mentioned

19. Which of the following is the correct output for the following JavaScript code?

```
varx=5,y=1
varobj={ x:10}
with(obj)
{
   alert(y)
}
```

(a) 10 (b) 1

(c) 5 (d) error

Answers

1. (a)	2. (d)	3. (b)	4. (d)	5. (a)	6. (a)	7. (d)	8. (c)	9. (d)	10. (a)
11. (c)	12. (a)	13. (d)	14. (b)	15. (c)	16. (a)	17. (d)	18. (c)	19. (b)	

Q. II Fill in the Blanks:

1. JavaScript is a computer programming language used as a part of web pages, whose implementations allow _____ script to interact with the user and make dynamic pages.
2. The jQuery _____ selector finds elements with a specific class.
3. Variables declared within a JavaScript function, become _____ to the function.
4. _____ provides capabilities for developers to create plug-ins on top of the JavaScript library.
5. JavaScript is a _____ -sensitive language means the identifiers Amar and AMAR will convey different meanings in JavaScript.
6. _____ is the most frequently used event type which occurs when a user clicks mouse left button.
7. jQuery is a very powerful tool which provides a variety of _____ traversal methods to help us select elements in a document randomly as well as in sequential method.
8. JavaScript _____ are symbols that are used to perform operations on operands. Take a simple expression 4 + 5 is equal to 9. Here 4 and 5 are called operands and '+' is called the operator.
9. With jQuery we select (query) _____ elements and perform "actions" on them.
10. JavaScript provides various _____ boxes to notify, warn, or to get input from the user.
11. To perform a multiway branching in JavaScript uses _____ statement.
12. The _____ loop is similar to the while loop except that the condition check happens at the end of the loop means that the loop will always be executed at least once, even if the condition is false.

13. jQuery is library of JavaScript file, containing all jQuery _____.
14. A function is a group of reusable code which can be called _____ in the program.
15. The _____ object lets us to work with a series of characters.
16. An HTML _____ can be something the browser does, or something a user does.
17. To define a function in JavaScript is by using the _____ keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.
18. A JavaScript function is executed when "something", _____ it (calls it).
19. The _____ statement "jumps over" one iteration in the loop.
20. jQuery _____ are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more.
21. The _____ (local and global) of a variable is the region of the program in which it is defined.
22. A _____ box is used to let the user make a choice. When Javascript pops up a confirm box, the user will have to click either "OK" or "Cancel" to proceed to the next step.
23. A return statement is used to specify the value/result/output that is _____ from a function.

Answers

1. client-side	2. .class	3. local	4. jQuery
5. case	6. onclick	7. DOM	8. operators
9. HTML	10. popup	11. switch	12. do...while
13. functions	14. anywhere	15. String	16. event
17. function	18. invokes	19. continue	20. selectors
21. scope	22. confirmation	23. returned	

Q. III State True or False:

1. JavaScript is the main client side scripting language that can be used to create, delete and manipulate HTML elements.
2. jQuery is a fast and concise JavaScript Library created by John Resig in 2006.
3. JavaScript can be implemented using JavaScript statements that are placed within the <script>... </script> HTML tags in a Web page.
4. JavaScript arithmetic operator like +, *, / etc., take operand (as a values or variable) and return the single value.
5. JavaScript is un-typed language means that a JavaScript variable can hold a value of any data type.
6. A function in JavaScript is declared with the var keyword.

7. An alert dialog box is mostly used to give a warning message to the users and has only one button "OK" to select and proceed.
8. A string can be converted to an array with the split() method.
9. String character is provided an index value starting from 1.
10. JavaScript lets us to execute code when events are detected.
11. The objective of a switch statement is to give an expression to evaluate and several different statements to execute based on the value of the expression.
12. jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.
13. The jQuery is not lightweight library.
14. A JavaScript function is a block of code designed to perform a particular task.
15. Event handlers can be used to handle and verify user input, user actions, and browser actions.
16. JavaScript supports dialog boxes which be used to raise and alert or to get confirmation on any input or to have a kind of input from the users.
17. The jQuery library harnesses the power of Cascading Style Sheets (CSS) selectors to let us quickly and easily access elements or groups of elements in the Document Object Model (DOM).
18. JavaScript supports all the features necessary to write modular code using functions.
19. The passed parameters can be captured inside the function and any manipulation can be done over those parameters.
20. In JavaScript function is a reusable codes that calls anywhere in the programs.
21. The slice() extracts a part of a string and returns the extracted part in a new string.
22. A confirmation dialog box displays a dialog box with two buttons OK and Cancel.
23. The #id selector in jQuery selects element whose class is match of given elements.

Answers

1. (T)	2. (T)	3. (T)	4. (T)	5. (T)	6. (F)	7. (T)	8. (T)	9. (F)	10. (T)
11. (T)	12. (T)	13. (F)	14. (T)	15. (T)	16. (T)	17. (T)	18. (F)	19. (T)	20. (T)
21. (T)	22. (T)	23. (F)							

Q. IV Answer the following Questions:

(A) Short Answer Questions:

1. What is JavaScript?
2. List any two features of JavaScript.
3. What is jQuery?
4. Enlist any two features of jQuey.

5. Define variable in JavaScript.
6. List data types in JavaScript.
7. Define regular expressions in JavaScript.
8. Define operator in JavaScript with example.
9. List control statement in JavaScript.
10. Define loop.
11. What is the purpose of switch statement in JavaScript?
12. What is the purpose of break and continue statements in JavaScript?
13. Define function in JavaScript.
14. Define events in JavaScript.
15. Define string in JavaScript.
16. List any two functions of string in JavaScript.
17. What is meant by popup boxes in JavaScript?
18. Give two ways to use jQuery.
19. Define selector in jQuery.
20. List any two DOM manipulation methods in jQuery.
21. How to call a jQuery library functions?
22. List any two CSS manipulation methods in jQuery.
23. List any two traversing DOM elements methods using jQuery.

(B) Long Answer Questions:

1. What is the purpose of JavaScript? What are its advantages and disadvantages?
2. What is variable? How to declare it? Explain with example.
3. Write short note on: Scope of variables.
4. What is the purpose of jQuery? List its advantages and disadvantages.
5. What data types used by JavaScript? Explain four of them in detail.
6. How to embedded JavaScript in program? Describe in detail.
7. What is operator in JavaScript? Enlist various types of JavaScript with example.
8. With the help of example describe if statement.
9. With the help of example explain if...else statement.
10. With the help of example describe switch statement.
11. What is loop? List its types.
12. Explain while and do-while loop with example. Also compare them.
13. Describe for and for-in loops. Differentiate them.
14. What is nested loop? Describe nested for loop with example.
15. Write JavaScript program to calculate area of circle.
16. What is break and continue statement? Explain them with example. Also differentiate between them.

17. What is function in JavaScript? How to create it? How to call it? Describe with example.
18. Write short note on: JavaScript HTML DOM events.
19. How to handle events in JavaScript? Explain JavaScript handler in detail
20. What is string in JavaScript? How to create it? Explain with example.
21. What are the popup boxes used in JavaScript? Explain Alert and Confirm popup boxes with example. Also compare them.
22. What is jQuery Library? How to it included in jQuery page.
23. What is jQuery selector? Explain #id and .class selectors with example. Also differentiate them with any two points.
24. How to manipulate DOM using jQuery? Describe in detail.
25. Differentiate between JavaScript and JQuery.

UNIVERSITY QUESTIONS AND ANSWERS

April 2016

1. State any two Window objects in JavaScript. **[1 M]**

Ans. Refer to Section 3.7.

2. Write the JavaScript data types. **[1 M]**

Ans. Refer to Section 3.3.1.

3. Explain alert dialog box in JavaScript with the help of suitable example. **[5 M]**

Ans. Refer to Section 3.11, point (1).

4. Write JavaScript code to open new window which will have some content op it and then close this Window. **[5 M]**

Ans. Refer to Additional Programs.

April 2017

1. What is the advantage of writing JavaScript in an external file? **[1 M]**

Ans. Refer to Section 3.3, Point (3).

2. What will be the return value for the following JavaScript statement?

```
typeof [1, 2, 3, 4]?
```

[1 M]

Ans. Refer to Additional Programs.

3. Write a JavaScript to accept a sentence from user and convert it to an array of words. Also display array element. **[5 M]**

Ans. Refer to Additional Programs.

4. Write a JavaScript function to validate user name and password against hardcoded values. **[4 M]**

Ans. Refer to Additional Programs.

5. What is the scope of variable in JavaScript? **[4 M]**

Ans. Refer to Section 3.3.3.

October 2017

1. Write the special operators used in JavaScript? **[1 M]**

Ans. Refer to Section 3.4.

2. Write a Javascript that accept a string from user. Pass this string as parameter to a function 'Check-Vowel' on button click event and return the count of the number of vowels within the string. **[5 M]**

Ans. Refer to Additional Programs.

3. Write a HTML form to accept student name, age and mobile no. from user. Using Javascript validate the following;

(a) Student name should not be empty

(b) Age must be in between 16 to 21. **[5 M]**

Ans. Refer to Additional Programs.

April 2018

1. List the special operators used in JavaScript. **[1 M]**

Ans. Refer to Section 3.4.

2. Write down the limitations of JavaScript. **[1 M]**

Ans. Refer to Section 3.1.1.2.

3. Explain the JavaScript confirm dialog box with suitable example. **[5 M]**

Ans. Refer to Section 3.11, Point (2).

4. State onmouseover and onkeypress JavaScript events. **[2 M]**

Ans. Refer to Section 3.8.

October 2018

1. What is the use of isnan() function in JavaScript? **[1 M]**

Ans. The isNaN() function determines whether a value is NaN (Not-a-Number) or not.

2. Discuss three kinds of pop up boxes in Java Script. **[5 M]**

Ans. Refer to Section 3.11.

3. Write Java script code to display clock in Text box (on load event). **[5 M]**

Ans. Refer to Additional Programs.

4. Which are the areas where Java Script is placed in the browser? Explain in detail. **[4 M]**

Ans. Refer to Additional Programs.

5. Explain any two Window object methods in JavaScript. **[2 M]**

Ans. Refer to Section 3.7.

April 2019

1. What are different types of object that JavaScript support? **[1 M]**

Ans. Refer to Section 3.7.

2. What are non-primitive Data types in JavaScript? **[1 M]**

Ans. Refer to Section 3.3.1.

3. Write any two advantages of JavaScript. **[1 M]**

Ans. Refer to Section 3.1.1.2.

4. What are different JavaScript control statements? Explain any two with example. **[5 M]**

Ans. Refer to Section 3.5.

5. Write a JavaScript code which will display patient master form having following fields patient Name, Addr, Contact No., Validate above fields with different criteria. **[5 M]**

Ans. Refer to Additional Programs.

6. Write a code using JavaScript to convert a string into uppercase. **[2 M]**

Ans. Refer to Additional Programs.

■■■

CHAPTER 4

AJAX

Objectives...

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

4.0 INTRODUCTION

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

4.1 OVERVIEW OF AJAX [April 16, 18, 19, Oct. 17]

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

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

- Ajax is a technique for creating fast and dynamic web pages.
- Ajax allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.
- Classic web pages, (which do not use Ajax) must reload the entire page if the content should change.
- Examples of applications using Ajax include Google Maps, Gmail, Youtube, and Facebook etc.

4.1.1 Advantages and Disadvantages of Ajax

Advantages of Ajax:

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

Disadvantages of Ajax:

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

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

4.2 AJAX WEB APPLICATION MODEL [April 16, 17, 18, 19]

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

1. Synchronous (Classic Web Application Model):

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

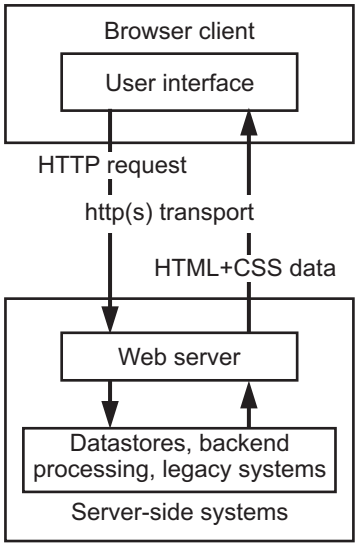


Fig. 4.1: Synchronous Web Application Model

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

2. Asynchronous (Ajax Web-Application Model):

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

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

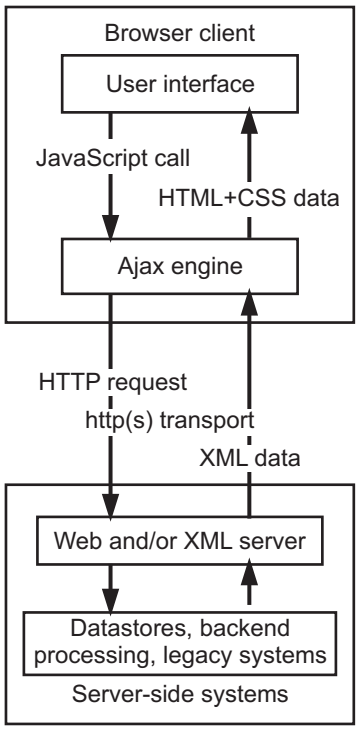


Fig. 4.2: Asynchronous Web Application Model

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

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

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

XMLHttpRequest Object's Properties:

1. **onreadystatechange**: An event handler for an event that fires at every state change. This property sets the method to be called on every state change.
2. **readyState**: The readyState property defines the current state of the XMLHttpRequest object.

   The following table provides a list of the possible values for the readyState property:

State	Description
0	The request is not initialized.
1	The request has been set up.
2	The request has been sent.
3	The request is in process.
4	The request is completed.

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

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

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

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

   (v) **readyState** = **4** After the request has been completed, and the response data has been completely received from the server.
3. **responseText**: Returns the response as a string.
4. **responseXM:** Returns the response as XML. This property returns an XML document object, which can be examined and parsed using the W3C DOM node tree methods and properties.
5. **status:** Returns the status as a number (e.g., 404 for "Not Found" and 200 for "OK").
6. **statusText:** Returns the status as a string (e.g., "Not Found" or "OK").

XMLHttpRequest Methods:

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

3. **getResponseHeader(headerName):** Returns the value of the specified HTTP header.
4. **open(method, URL)**
   open(method, URL, async)
   open(method, URL, async, userName)
   open(method, URL, async, userName, password)
   Specifies the method, URL, and other optional attributes of a request.
   The method parameter can have a value of "GET", "POST", or "HEAD". Other HTTP methods, such as "PUT" and "DELETE" (primarily used in REST applications) may be possible.
   The "async" parameter specifies whether the request should be handled asynchronously or not. "true" means that the script processing carries on after the send() method without waiting for a response, and "false" means that the script waits for a response before continuing script processing.
5. **send(content):** Sends the HTTP request to the server and receives the response when the readyState value is 1.
6. **setRequestHeader(label, value):** Adds a label/value pair to the HTTP header to be sent.

Creating the XMLHttpRequest Object:

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

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

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

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

- The following code is used to create XMLHttpRequest object:

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

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

How XMLHttpRequest Object Works in Synchronous Pages Pattern:

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

How XMLHttpRequest Object Works in Asynchronous Pages Pattern:

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

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

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

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

Output:

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

4.3 AJAX - PHP FRAMEWORK [Oct. 14]

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

Sajax Framework Example:

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

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

  The part of mult.php:

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

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

The part of mult.php:

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

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

4.4 PERFORMING AJAX VALIDATION

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

a.html

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

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

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

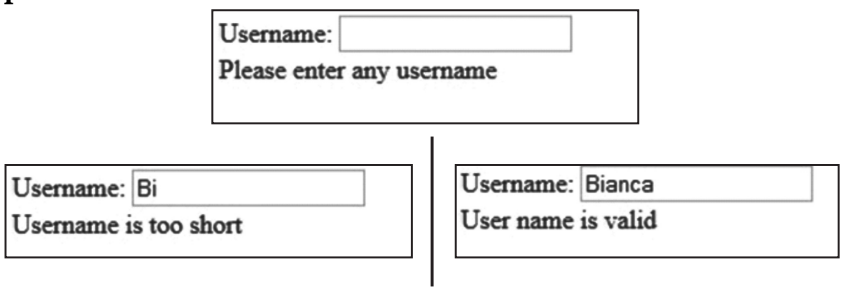
  `http://localhost/a.html`

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

validate.php

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

Output:



4.5 HANDLING XML DATA USING PHP AND AJAX

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

index.html

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

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

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

bookstore.xml

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

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

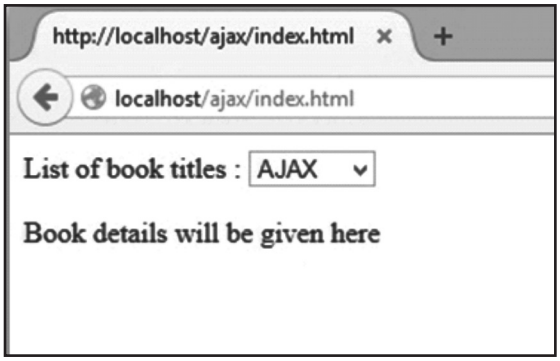
- The XMLHttpRequest method creates URL to request getbook.php file. This URL has parameter q which is used to initialize the value of list box and then sends the request to getbook.php page. After completion of response by the server, the stateChanged method is called.
- The server page getbook.php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

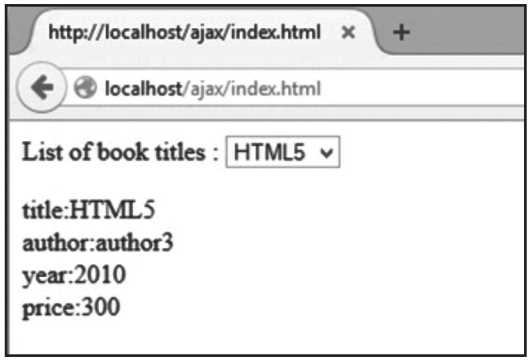
getbook.php

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

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

Output:





4.6 CONNECTING DATABASE USING PHP AND AJAX

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

db.html

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

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

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

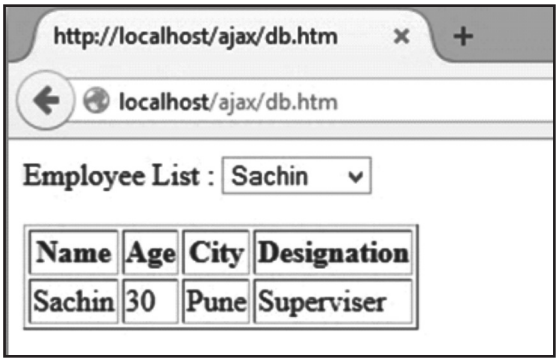
getemployee.php

```
<?php
   q=$_GET["q"];
      $conn = pg_pconnect("dbname=test") or die("An error occurred.");
       $result = pg_query($conn, "SELECT * FROM employee WHERE id = ". $q)
                                              or die("An error occurred.");
      echo "<table border='1'>
      <tr>
         <th>Name</th>
         <th>Age</th>
```

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

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

Output:



ADDITIONAL PROGRAMS

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

Book_list.php

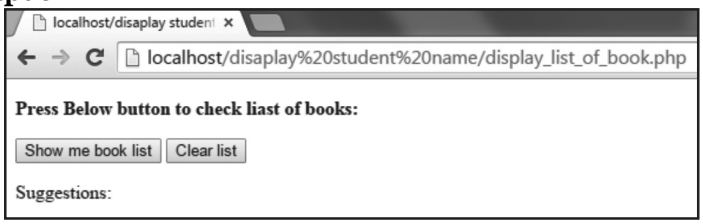
```
<html>
    <head>
        <script>
            function showMatch()
            {
            //var str = document.getElementById("search_string");
            /*if (str.length==0)
            {
                document.getElementById("txtHint").innerHTML="";
                return;
            }*/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

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

Getmatch.php

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

Output:



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

New.php

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

```
            function showHint(str)
            {
               if(str.length == 0)
               {
                   document.getElementById("txtHint").innerHTML = "";
                   return;
               }
               else
               {
                   var xmlhttp = new XMLHttpRequest();
                   xmlhttp.onreadystatechange = function()
                   {
                      if(xmlhttp.readyState == 4 && xmlhttp.status == 200)
                      {
                         document.getElementById("txtHint").innerHTML =
                                                        xmlhttp.responseText;
                      }
                   }
                      xmlhttp.open("GET", "gethint.php?q=" + str, true);
                      xmlhttp.send();
                   }
            }
         </script>
      </head>
      <body>
         <p><b>Start typing a name in the input field below:</b></p>
         <form>
            First name: <input type="text" onkeyup="showHint(this.value)">
         </form>
            <p>Suggestions: <span id="txtHint"></span></p>
      </body>
   </html>
```

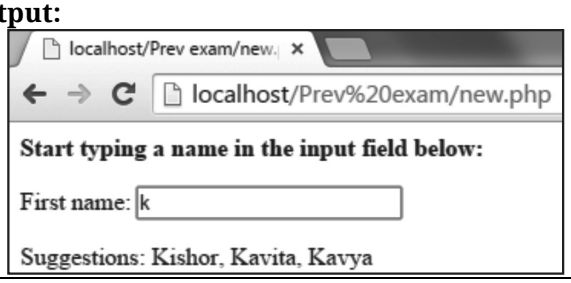
Gethint.php

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

```
$a[] = "Bob";
$a[] = "Chinmay";
$a[] = "Deepa";
$a[] = "Esha";
$a[] = "Faiz";
$a[] = "Gautam";
$a[] = "Harshal";
$a[] = "Ishant";
$a[] = "Jyotsana";
$a[] = "Kishor";
$a[] = "Lokesh";
$a[] = "Nina";
$a[] = "Omkar";
$a[] = "Pritam";
$a[] = "Amey";
$a[] = "Rahul";
$a[] = "Chaitali";
$a[] = "Dushyant";
$a[] = "Kavita";
$a[] = "Kavya";
$a[] = "Sunil";
// get the q parameter from URL
$q = $_REQUEST["q"];
$hint = "";
// lookup all hints from array if $q is different from ""
if($q !== "")
{
   $q = strtolower($q);
   $len=strlen($q);
   foreach($a as $name)
   {
      if(stristr($q, substr($name, 0, $len)))
      {
          if ($hint === "")
          {
              $hint = $name;
```

```
                } else {
                    $hint .= ", $name";
                }
            }
        }
    }
    // Output "no suggestion" if no hint was found or output correct values
    echo $hint === "" ? "no suggestion" : $hint;
?>
```

Output:



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

Disp_list.php

```
<html>
    <head>
        <script>
            function showMatch()
            {
            //var str = document.getElementById("search_string");
            /*if (str.length==0)
            {
                document.getElementById("txtHint").innerHTML="";
                return;
            }*/
            if(window.XMLHttpRequest)
            { // code for IE7+, Firefox, Chrome, Opera, Safari
                xmlhttp=new XMLHttpRequest();
            }
            else
            { // code for IE6, IE5
                xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
            }
```

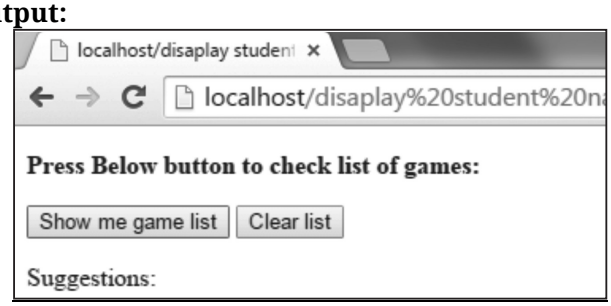
```
            xmlhttp.onreadystatechange=function()
        {
            if(xmlhttp.readyState==4 && xmlhttp.status==200)
            {
                document.getElementById("txtHint").innerHTML=
                                                xmlhttp.responseText;
            }
            }
 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

Getmatch.php

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

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

Output:



PRACTICE QUESTIONS

Q. I Multiple Choice Questions:

1. AJAX stands for,
   (a) Asynchronous JavaScript And XML (b) Automatic JavaScript And XML
   (c) Angular JavaScript And XML (d) Analogous JavaScript And XML
2. Which object is an API in the form an object whose methods help in transfer of data between a web browser and a web server?
   (a) HTMLHttpRequest (b) XMLHttpRequest
   (c) ClientBrowserHttpRequest (d) XMLJavaHttpRequest

3. Ajax is a group of inter-related technologies like
   (a) HTML/XHTML (b) DOM and XML
   (c) JavaScript and CSS (d) All of the mentioned
4. Ajax can be used for interactive communication with an,
   (a) XML file (b) HTML file
   (c) CSS file (d) JavaScript file
5. Ajax is used for creating,
   (a) Desktop applications (b) Mobile applications
   (c) Web applications (d) System applications
6. Ajax sends data to a web server,
   (a) before loading the page (b) in the background
   (c) with reloading the page (d) Zephyr
7. Following which frameworks used to integrate Ajax with PHP?
   (a) CakePHP (b) Sajax
   (c) Zephyr (d) All of the mentioned
8. Which is an Ajax based framework which generates Ajax -enabled JavaScript from many server side languages like PHP, ASP, ColdFusion, Perl, Python and so on?
   (a) Xajax (b) Tajax
   (c) Sajax (d) Pajax
9. Which property holds the status of the XMLHttpRequest?
   (a) readyState (b) holdState
   (c) pauseState (d) stopState

Answers

1. (a)	2. (b)	3. (d)	4. (a)	5. (c)	6. (b)	7. (d)	8. (c)	9. (a)	

Q. II Fill in the Blanks:

1. Ajax is a technology for developing better, faster and interactive _____ Applications using HTML, CSS, JavaScript and XML.
2. The _____ is used for asynchronous communication between client and server.
3. Ajax allows us to send and receive data asynchronously _____ reloading the web page.
4. _____ is stands for Asynchronous JavaScript And XML.
5. All modern browsers (Chrome, Firefox, Edge (and IE7+), Safari, Opera) have a built-in XMLHttpRequest object to make HTTP _____.
6. Storing the data into database and retrieving it from database, we can do this using PHP and _____.
7. Ajax is a _____ Web technique.

8. Ajax is used to create more interactive applications with _____.
9. The _____ object can be used in calling the Web page in either synchronous or asynchronous mode.

Answers

1. Web	2. XMLHttpRequest	3. without	4. AJAX
5. requests	6. PostgreSQL	7. client-side	8. PHP
9. XMLHttp			

Q. III State True or False:

1. Ajax technologies includes HTML (used for defining the structure of a Web application, CSS (used to provide look and style to a Web application), JavaScript (used for making a Web application interactive, interesting and user friendly), XML (a format to store and transport data from the Web Server) and so on.
2. Ajax is a programming language.
3. XMLHttpRequest provides an easy way to retrieve data from web server.
4. The XMLHttpRequest object can be used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.
5. AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.
6. Ajax is a server-side Web technique.
7. Syntax for creating an XMLHttpRequest object in Ajax is, variable = new XMLHttpRequest();
8. Ajax can be used for interactive communication with a database.
9. An object of XMLHttpRequest is used for asynchronous communication between client and server.
10. An asynchronous request block the client i.e. browser is responsive.

Answers

1. (T)	2. (F)	3. (T)	4. (T)	5. (T)	6. (F)	7. (T)	8. (T)	9. (T)	10. (F)

Q. IV Answer the following Questions:

(A) Short Answer Questions:

1. Give full form of Ajax?
2. “Ajax is not a technology”. Comment this statement.
3. List applications of Ajax.
4. What is the purpose of XMLHttpRequest object?
5. What is Sajax?

(B) Long Answer Questions:

1. What is Ajax? State its advantages and disadvantages.
2. With the help of diagram describe Ajax Web application model.
3. How to handling XML data using PHP and Ajax.
4. What is XMLHttpRequest object? How to create it? Expalin in detail.
5. Write a short note on: Ajax-PHP framework.
6. How to perform validations using Ajax? Explain with example.
7. With the help of diagram describe how to connecting a database using PHP and Ajax.
8. What are the technologies used by Ajax? Explain four of them.
9. How XMLHttpRequest object works in Synchronous and Asynchronous pages pattern? Describe in detail.

UNIVERSITY QUESTIONS AND ANSWERS

April 2016

1. How we can send the data to server using Ajax? **[1 M]**

Ans. Refer to Section 4.1.

2. Which object is Ajax make Web page interactive? **[1 M]**

Ans. Refer to Section 4.2.

3. Write an Ajax program to search the student name according the character typed and display list using array. **[5 M]**

Ans. Refer to Additional Programs.

4. Write note on Ajax web application model. **[5 M]**

Ans. Refer to Section 4.2.

April 2017

1. How to get response from XMLHTTPRequest object received from server in Ajax? **[1 M]**

Ans. Refer to Section 4.2.

2. Differentiate between synchronous and asynchronous request to the server in Ajax. Explain with example. **[5 M]**

Ans. Refer to Section 4.2.

3. Accept cricket player name from user. Write a program to search player name into cricket.xml file and display details (name, runs, wickets) using Ajax (use DOM functions). **[5 M]**

Ans. Refer to Additional Programs.

October 2017

1. Give any two applications of Ajax. **[1 M]**

Ans. Refer to Section 4.1.

2. Explain the working of Ajax in detail. **[5 M]**

Ans. Refer to Section 4.2.

April 2018

1. Write any two Ajax applications. **[1 M]**

Ans. Refer to Section 4.1.

2. Draw and explain Ajax web application model. **[5 M]**

Ans. Refer to Section 4.2.

3. Write an Ajax program to search student name according to the character typed and display same list using array. **[5 M]**

Ans. Refer to Additional Programs.

October 2018

1. What is the need of XMLHttpRrequest object in Ajax? **[1 M]**

Ans. Refer to Section 4.2.

April 2019

1. What are different values of read State property of XMLHttpRequest? **[1 M]**

Ans. Refer to Section 4.2.

2. What is an AJAX? **[1 M]**

Ans. Refer to Section 4.1.

3. Write an AJAX program to display list of game stored in an array on click ok Button. **[5 M]**

Ans. Refer to Additional Programs.

4. Write a note on Ajax Web Application model. **[4 M]**

Ans. Refer to Section 4.2.

■■■

CHAPTER 5

PHP Framework CodeIgniter

Objectives...

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

5.0 INTRODUCTION

- PHP development is on the rise and more than 50% of web development mostly preferred PHP.
- PHP is an open source, freely available scripting language, ability to work on various OS (Operating System) and ease in integration with several types of databases.
- MVC framework adds up to the beauty of PHP programming and makes it easier for the developer to build robust programs using the classic features of MVC framework.
- However, there are dozens of MVC frameworks at the disposal which can add up to confusion. Therefore, choosing the right PHP framework can be a really daunting task at times.
- Fig. 5.1 shows MVC Framework of PHP.

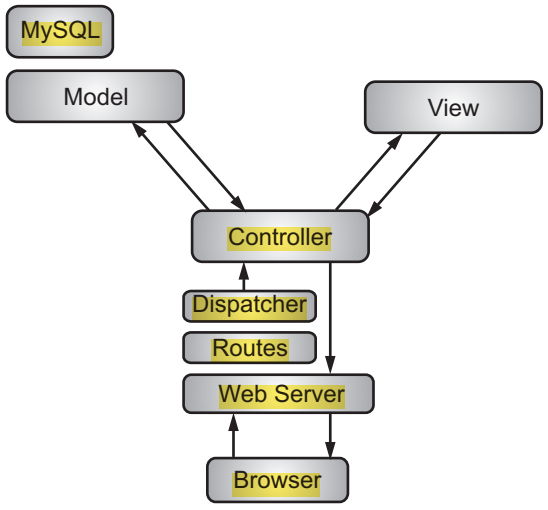


Fig. 5.1: Model View Controller (MVC) Framework

- The Model View Controller architecture primarily reduces the burden of the developers by providing a lucid and presentable coding pattern.
- Its segregation into different parts makes it easier for the programmer to write structured and neat code.
- The segmented feature helps a new developer in understanding the codes easily and also helps in tracing the bugs easily.
- In MVC model:
  1. **Model (M):** This is the one who knows all about data and databases. It is responsible to represent the data in the application.
  2. **View (V):** This section is in-charge of displaying the data to the user. It takes inputs from the Model to display the desired output to the user.
  3. **Controller (C):** Controller acts as an interface between the View and the Model and commands the Model to perform the appropriate action as requested by the user.
- For building a Web application we spend a lot of time in writing the same code again and again. Frameworks provide a starting block and minimize the amount of code needed to build a website.
- CodeIgniter is PHP driven framework. CodeIgniter is a powerful PHP framework with a very small footprint, built for developers who need a simple and elegant toolkit to create full-featured web applications.
- CodeIgniter 4 is the current version of the framework, intended for use with PHP 7.3+ (including 8.0). Development is ongoing, and the current version is v4.1.4.
- CodeIgniter contains libraries, simple interface and logical structure to access these libraries, plug-ins, helpers and some other resources which solve the complex functions of PHP more easily maintaining a high performance.
- It simplifies the PHP code and brings out a fully interactive, dynamic website at a much shorter time.

5.1 OVERVIEW OF CODEIGNITER

- CodeIgniter is an open-source software rapid development web framework, for use in building dynamic web sites with PHP. CodeIgniter is a PHP MVC framework used for developing Web applications.
- CodeIgniter provides out of the box libraries for connecting to the database and performing various operations like sending emails, uploading files, managing sessions, etc.
- CodeIgniter is a powerful PHP framework with a very small footprint, built for developers who need a simple and elegant toolkit to create full-featured web applications.

Features of CodeIgniter:

1. **MVC Architecture:** The PHP CodeIgniter framework uses the Model-View-Controller (MVC) architectural design. The MVC architecture is an industry standard practice when working with web applications. MVC architecture separates the data, business logic, and presentation.
2. **Extendable:** CodeIgniter comes with some libraries and helpers out of the box. If what we want is not there or we would like to implement an existing feature the way. Then we can do so easily by creating our libraries, helpers, packages, etc. We can also create REST API in CodeIgniter.
3. **Loosely Coupled:** The built-in features of CodeIgniter are designed to work independently without relying too much on other components which makes it easy to maintain and make upgrades
4. **Small Footprint:** The entire source code for CodeIgniter framework is close to 2MB. This makes it easy to master CodeIgniter and how it works. It also simplifies deploying and updating it.
5. **Excellent Documentation:** The CodeIgniter framework is well documented, and there are good books, tutorials and answered forum questions on CodeIgniter. This means whatever challenge that we have, chances are someone has already encountered the problem, solved it and the solution is out there for you.
6. **Application specific Built-in Components:** CodeIgniter has components for sending email, database management, session management and many more.
7. **Blazing Fast:** In CodeIgniter users tend to favor applications that load very fast.

How CodeIgniter Works?

- CodeIgniter framework is mainly driven by the technology of PHP which contains resources like plug-ins, helpers, and libraries.
- With the help of these resources, the complex functions and procedures of the framework are tackled cleverly by the incredible technology of PHP.
- CodeIgniter is an MVC framework. MVC stands for Model View Controller. When a user requests a resource, the Controller responds first.
- The controller understands the user request then request the necessary data if necessary.
- For example, if we want to retrieve a customer with the id = 3, the controller will receive the request, then request the CodeIgniter models to retrieve the record with the id of 3.
- The CodeIgniter models will return the record to the controller. The controller then forwards the result to the view which formats it into a human-readable format and then the results are returned to the user in the browser.

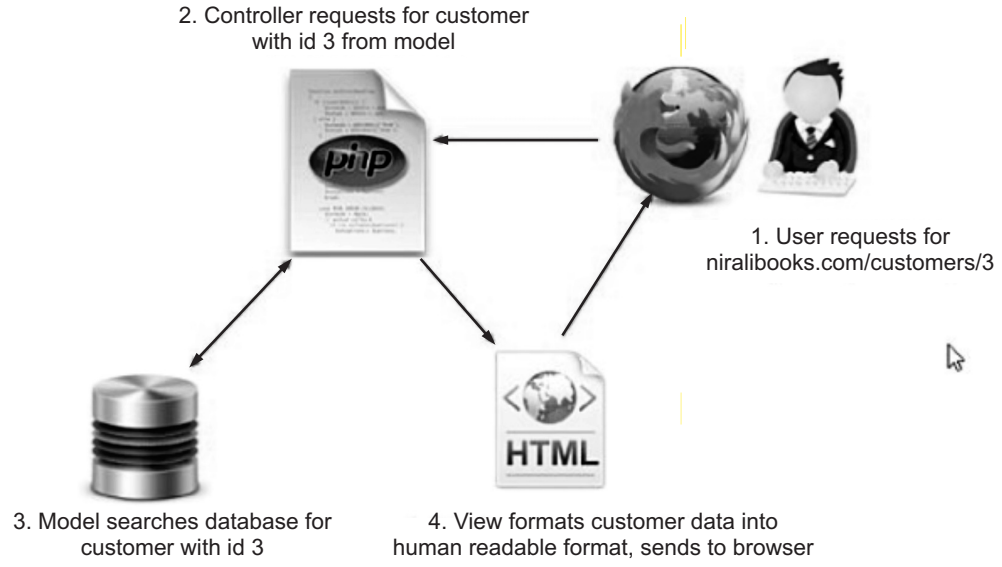


Fig. 5.2

5.1.1 Advantages and Disadvantages of CodeIgniter

Advantages of CodeIgniter:

1. The CodeIgniter application development requires less effort and time because of the strong and clear documentation that the PHP programmers can easily develop a range of web applications which can be easily set up.
2. CodeIgniter framework makes it easier for software developers to write secure PHP applications by giving various features. CodeIgniter help software developers/engineers to shield the web application from basic security dangers like SQL infusion, cross website scripting assaults, and remote code execution.
3. CodeIgniter gives user-friendly UI (User Interface) that the technology offers. It application development is made smooth, dynamic, secure and flexible by getting the most out of this intuitive user interface.
4. CodeIgniter has rich set of libraries which are easy to write and change it's behavior.
5. Customization is the key to building scalable, feature-rich and highly engaging web solutions. Due to CodeIgniter's extensibility, PHP developers can channel CodeIgniter framework resources towards customized CodeIgniter application development.
6. CodeIgniter offers flexibility and easy management with MVC based framework.
7. The easy nature of this framework allows the users to configure the common tasks related to web development.
8. Moreover, the convenience of UI also adds to the speed of app development, effectively reducing the overall time span for larger web application projects.

Disadvantages of CodeIgniter:

1. **Lack of Libraries:** CodeIgniter framework has fewer tools and built-in libraries when compared with other frameworks. The CodeIgniter Web application development allows certain customization by which a number of libraries can be provided. In this framework, the developers can only install the libraries which are necessary for the Web application.
2. **Rigorous Adherence to Naming Conventions:** CodeIgniter application development suffers from a hideous limitation when it comes to naming files and folder. The CodeIgniter framework lacks modern namespace and autoloader use, which means, it uses a standard approach for folder and file naming convention. As developers are required to adhere to this, it limits flexibility to an extent.
3. **Absence of Default Modular Separation of Code:** CodeIgniter development organization does not make structured and readable codes because of the high quality that the codes can be updated, modified and maintained according to the future requirements. As CodeIgniter framework does not support the modular separation, the programmers need to give extra effort and time for the maintenance of the code.

5.2 INSTALLING CODEIGNITER

- CodeIgniter is an application development framework, which can be used to develop websites, using PHP.
- CodeIgniter is an Open Source framework. CodeIgniter has a very rich set of functionality, which will increase the speed of website development work.
- The source code for the CodeIgniter Framework is available on the official CodeIgniter website.
- If we want to download the latest version of the framework, then you should do it from the official web page.

Step 1 : **Download CodeIgniter Framework:** Open the following URL in the browser https://codeigniter.com/ then download link to the latest version of the framework.

Step 2 : **Unzip CodeIgniter-4.1.4.zip File:** Clicking the above link will download the framework as a zipped folder and Unzip the contents of CodeIgniter-4.1.4.zip.

Step 3 : **Create a new directory:** Let's say you want to create a project called the online store. We can follow the following steps to start the project. Create a new directory on the development drive, e.g, D:\Sites\online-store.

Step 4 : **Open the contents of CodeIgniter-4.1.4:** Now, we should be able to see the following files. Copy the above contents to the project directory, e.g., D:\Sites\online-store.

Step 5 : Open the Terminal and Run the following command. Start the built-in PHP server, just to make sure everything is OK.

```
cd D:\Sites\ online-store
```

Run the following command,

```
php -S localhost:3000
```

Step 6 : Open the below URL (See Fig. 5.3). Load the following URL into the browser. http://localhost:3000/

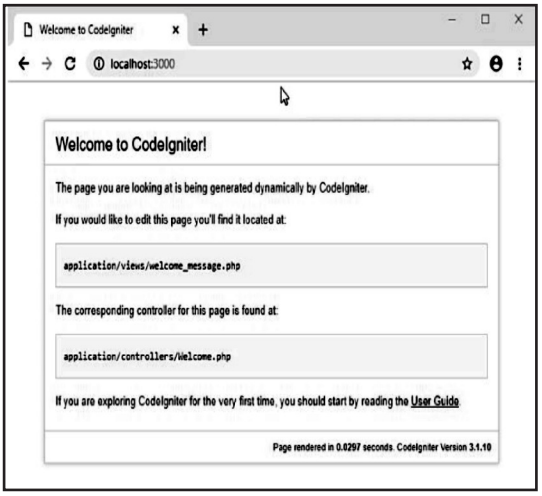


Fig. 5.3

- The Composer in CodeIgniter is a package management system for PHP. A package is simply a collection of PHP scripts that work together towards a single goal/objective.
- Based on this definition, CodeIgniter can even though it's a framework, qualifies to be labeled a package in composer terminologies.

5.3 APPLICATION ARCHITECTURE

- CodeIgniter is open source software, licensed under MIT License and its source code is maintained at GitHub.
- The architecture of CodeIgniter application is shown in Fig. 5.4. The important steps in the workflow are explained below:
  1. The index.php is the startup file of CodeIgniter which contains the code for initializing the basic resources required for the smooth execution of the application.
  2. The routing decides how to route the information further.
  3. If the request file exists in cache folder, then directly the response is sent to the browser, else the process continues to security check.
  4. Before loading the controller, the HTTP request along with the submitted data is passed to the security check.

5. The application controller loads the models, views, helpers, libraries, plugins etc. as demanded by the request.
6. The view is generated and cached for future use.
7. The view is then sent to the browser as a response.

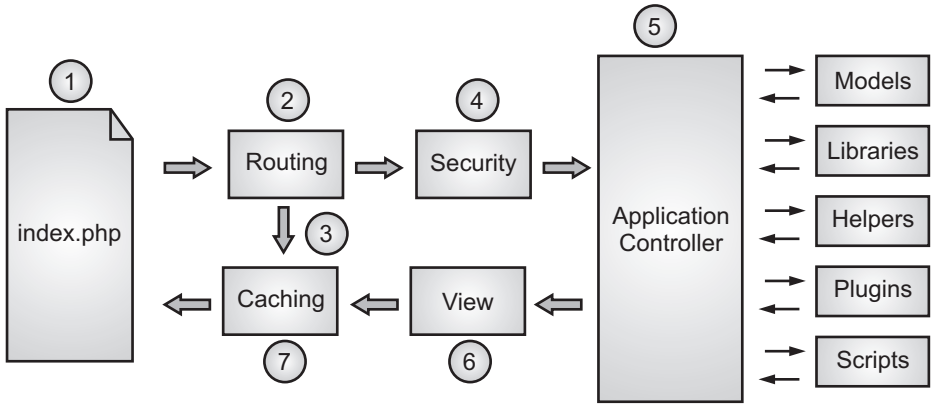


Fig. 5.4: Application Architecture of CodeIgniter

- CodeIgniter directory structure is divided into following three folders:
  - **Application:** This is the directory that will contain your application logic. All of our application code will be contained in this directory.
  - **System:** This folder contains the framework core files. It is not advised to make changes in this directly or put our own application code into this directory.
  - **user_guide:** This directory contains the user manual for CodeIgniter.
- Let us see above components/folders in detail:

1. Application:

- As the name indicates the Application folder contains all the code of the application that we are building. This is the folder where we will develop our project.
- Application folder contains the main components of an application, such as model, controller and view, important configuration files etc. and as such is the main development folder. Application folder contains application logic.
- The Application folder contains several other folders, which are explained below:–
  - **Cache** folder contains all the cached pages of the application. These cached pages will increase the overall speed of accessing the pages.
  - **Config** folder contains various files to configure the application. With the help of config.php file, we can configure the application. Using database.php file, we can configure the database of the application.
  - **Controllers** folder holds the controllers of the application. It is the basic part of the application.
  - **Core** folder will contain base class of your application.

- In **Helpers** folder, we can put helper class of the application.
- The files in **Hooks** folder provide a means to tap into and modify the inner workings of the framework without hacking the core files.
- **Language** folder contains language related files.
- **Libraries** folder contains files of the libraries developed for your application.
- **Logs** folder contains files related to the log of the system.
- **Models** folder is the database login will be placed in this folder.
- In T**hird_party** folder, we can place any plugins, which will be used for the application.
- In **Views** folder the application's HTML files will be placed.

2. System:

- This folder contains CodeIgniter core codes, libraries, helpers and other files, which help make the coding easy. These libraries and helpers are loaded and used in web app development.
- The system folder contains all the CodeIgniter code of consequence, organized into following various folders:
  - The **Core** is folder contains CodeIgniter's core class.
  - The **Database** folder contains core database drivers and other database utilities.
  - The **Fonts** folder contains font related information and utilities.
  - The **Helpers** folder contains standard CodeIgniter helpers (such as date, cookie, and URL helpers).
  - The **Language** folder contains language files. You can ignore it for now.
  - The **Libraries** folder contains standard CodeIgniter libraries.

3. user_guide:

- This is user guide to CodeIgniter. It is basically, the offline version of user guide on CodeIgniter website.
- Using user_guide, one can learn the functions of various libraries, helpers and classes. It is recommended to go through this user guide before building the first web app in CodeIgniter.
- Beside these three folders, there is one more important file named "index.php". In this file, we can set the application environment and error level and we can also define system and application folder name.

5.4 MVC FRAMEWORK

- CodeIgniter is based on the Model-View-Controller (MVC) development architecture. MVC standards for Model-View-Controller.
- MVC is a software design pattern which separates application logic from its presentation which dictates the clear separation of concerns.

- In MVC, interface elements are never invoked directly, they are always loaded from the controller.
- CodeIgniter uses the Model, View, Controller (MVC) pattern to organize the files. MVC is an architectural pattern that splits the application into following three major components:
  1. **Model**: The MVC model represents application's data structures. Model classes interface with database and contain functionality for retrieving, inserting and updating information in the database.
  2. **View:** It is responsible for data presentation. A view contains information presented to an end user and on most occasions is normally a Web page.
  3. **Controller:** It is an interface between a model and a view. In MVC all user requests are handled by a controller. It coordinates the activities between the model and the view.
- MVC is a software approach that separates application logic from presentation. In practice, it permits the Web pages to contain minimal scripting since the presentation is separate from the PHP scripting.
- Fig. 5.5 shows the interaction between model, view and controller in MVC architecture.

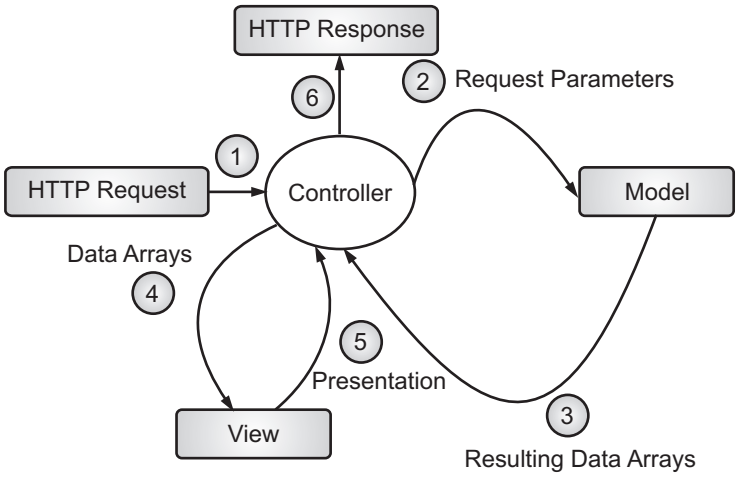


Fig. 5.5: The Workflow in MVC Architecture

- The steps in Fig. 5.5 shows following steps:
  1. Browser sends HTTP request to a controller.
  2. The controller retrieves the request parameters and instantiates the model, sets the model parameters and optionally invokes one or more model functionalities. The result of this is the generation of a data array.
  3. The data arrays resulting from interaction of controller with the model are passed on the view.
  4. The view employs this data for generating presentation.
  5. Finally, the presentation generated by a view is sent to the browser as a response.

Real-Word Example for MVC Architecture:

- The following steps elaborate application of MVC architecture to User Registration Application:

  Step 1: Controller receives user request for registration in the database.

  Step 2: The validation which does not require interaction with the database is carried out by the controller.

  Step 3: If the validation requires database interaction, the request is delegated to the model and the controller waits for the response from the model. The validation requiring any database interaction is carried on server side.

  Step 4: If the validation fails, the user is redirected to the registration page with appropriate error messages.

  Step 5: On successful validation, the data is submitted to the model for persisting in the database and waits for the response from the model.

  Step 6: On receiving the response from the model, the controller loads the appropriate view and passes the data retuned by model as parameter.

- Fig. 5.6 shows the interaction between model, view and controller in user registration application:

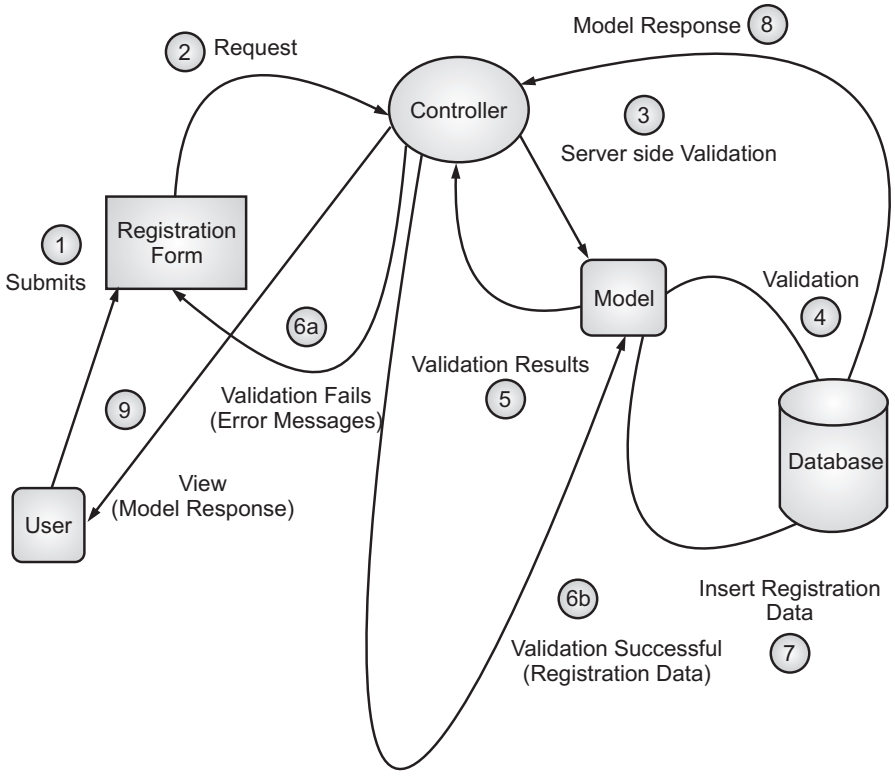


Fig. 5.6

Advantages of MVC Architecture:

1. **Loose Coupling:** In MVC the components exist and function independently of each other.
2. **Flexibility:** In MVC one can easily make changes to individual components.
3. **Increased Productivity:** In MVC architecture more than one person can work on the project at the same time. The front-end developers can work on views and presentation.

5.5 BASIC CONCEPT OF CODEIGNITER

- Codelgniter is an extremely light weight, high performance and open source PHP framework for developing Websites in Model-View-Controller architecture.
- Codelgniter possesses a rich set of functionality with numerous libraries for sending e-mails, uploading files, connecting to database, session tracking etc. and numerous helpers which tremendously increase the speed of Website development in PHP thereby contributing to the reduction in development cost and time.
- A software framework is a blueprint or an abstraction incorporating a generic functionality which can be overridden by application specific functionality enabling rapid application development.
- A software framework provides a standard way for building and deploying applications. It makes the code easier to read and maintain.
- The prominent characteristics of CodeIgniter are explained below:
  1. Codelgniter is licensed under MIT license with a source code maintained in GitHub, so it is free to use.
  2. Codelgniter is extremely lightweight. Codelgniter core system library is extremely lightweight, other libraries are dynamically loaded on request.
  3. Codelgniter adopts Model-View-Controller (MVC) design pattern which separates application logic from its presentation.
  4. Codelgniter targets to deliver the maximum performance in a minimal time with the help of a minimal code.
  5. Codelgniter has rich set of libraries for database management, creating and validating a form, sending e-mail, file uploading etc.
  6. Codelgniter has component singularity feature. Each component has a focused objective and implements functionality to achieve the stated purpose.
  7. Codelgniter generates URLs which are search engine friendly since it employs segment-based approach in contrast to query-based approach.

Concept of Models in MVC:

- Models in CodeIgniter are the classes are designed to work with information in the database.

- Example, if we are using CodeIgniter to manage users in the application then we must have model class, which contains functions to insert, delete, update and retrieve the users' data.

Creating Model Class:

- In CodeIgniter Model classes are stored in application/models directory. Following program code shows how to create model class in CodeIgniter.

```
<?php
   class Model_name extends CI_Model
   {
      Public function __construct()
      {
         parent::__construct();
      }
   }
?>
```

Where, Model_name is the name of the model class that we want to give. Each model class must inherit the CodeIgniter's CI_Model class. The first letter of the model class must be in capital letter.

- Following is the program code for users' Model class:

```
<?php
   class User1_model extends CI_Model
   {
      public function __construct()
      {
         parent::__construct();
      }
   }
?>
```

- The above model class must be saved as User1_model.php. The class name and file name must be same.

Loading Model:

- Model can be called in controller. Following program code can be used to load any model.

```
$this->load->model('model_name');
```

Where, model_name is the name of the model to be loaded and after loading the model we can simply call its method as shown below:

```
$this->model_name->method();
```

Auto-loading Models:

- There may be situations or conditions where we want some model class throughout our application. In such situations, it is better if we autoload it.
- As given below, pass the name of the model in the array that we want to autoload and it will be auto-loaded, while system is in initialization state and is accessible throughout the application.

```
/*
| -------------------------------------------------------------------
|  Auto-Load Models
| -------------------------------------------------------------------
| Prototype:
|
|   $autoload['model'] = array('first_model', 'second_model');
|
| We can also supply an alternative model name to be assigned
| in the controller:
|
|   $autoload['model'] = array('first_model' => 'first');
*/
$autoload['model'] = array();
```

- Helpers in CodeIgniter will help us to build the system. A number of helpers are available in CodeIgniter while we can build our own helpers too. A helper can be loaded as, `$this->load->helper('name');`
  where, name is the name of the helper. For example, if we want to load the URL Helper, then it can be loaded as, `$this->load->helper('url');`
- CodeIgniter has user-friendly URI routing system, so that we can easily re-route URL. We will find an array called $route in which we can customize the routing rules.
- We can also use regular expressions in $route array key part. If any URI matches with regular expression, then it will be routed to the value part set into $route array.

Concept of Controllers in MVC:

- A controller in CodeIgniter is a simple class file. As the name suggests, controller controls the whole application by URI.
- In CodeIgniter, a Controller is simply a class file that is named in a way that it can be associated with a URI.

Creating a Controller:

- For creating controller in CodeIgniter we first, go to application/controllers folder. We will find two files there, index.html and Welcome.php. These files come with the CodeIgniter.

- Keep these files as they are. Create a new file under the same path named “Test1.php” write the following program code in that file:

```
<?php
   class Test1 extends CI_Controller
   {
      public function index()
      {
         echo "Hello World!";
      }
   }
?>
```

- The Test1 class extends an in-built class called CI_Controller. This class must be extended whenever we want to make your own Controller class.

Calling a Controller:

- The above Controller can be called by URI as given below:

```
http://www.your-domain.com/index.php/test1
```

- We notice the word “test1” in the above URI after index.php. This indicates the class name of controller. As we have given the name of the controller “Test1”, we are writing “test1” after the index.php.
- The class name must start with uppercase letter but we need to write lowercase letter when we call that controller by URI.
- The general **syntax** for calling the controller is as given below:

```
http://www.your-domain.com/index.php/controller/method-name
```

Creating and Calling Constructor Method:

- Let us modify or update the above class and create another method named “Display1”.

Test.php

```
<?php
   class Test1 extends CI_Controller
     {
     public function index()
     {
         echo "The CodeIgniter default Function.";
      }
      public function Display1()
     {
         echo "This is Display function.<br>";
         echo " Its written in Display method. ";
      }
   }
?>
```

- We can execute the above controller in the following three ways:

  ```
  http://www.your-domain.com/index.php/test
  http://www.your-domain.com/index.php/test/index
  http://www.your-domain.com/index.php/test/hello
  ```

  - After visiting the first URI in the browser, we get the output is `The CodeIgniter default Function`. As we can see, we got the output of the method "index", even though we did not pass the name of the method the URI. We have used only controller name in the URI. In such situations, the CodeIgniter calls the default method "index".
  - Visiting the second URI in the browser, we get the same output as `The CodeIgniter default Function`. Here, we have passed method's name after controller's name in the URI. As the name of the method is "index", we are getting the same output.
  - Visiting the third URI in the browser, we get the following output:

    ```
    This is Display function.
    Its written in Display method.
    ```

  - As we can see, we are getting the output of the method "hello" because we have passed "hello" as the method name, after the name of the controller "test" in the URI.
- The name of the controller class must start with an uppercase letter. The controller must be called with lowercase letter.

Concept of Views in MVC:

- Views can be a simple or complex Webpage, which can be called by the controller. The Webpage may contain header, footer etc.
- View cannot be called directly. Let us create a simple view in CodeIgniter. We can create a new file under application/views with name "test1.php".

```
<html>
   <head>
      <meta charset = "utf-8">
      <title> Creating View in CodeIgniter </title>
   </head>
   <body>
       This example shows how to execute View in CodeIgniter!
   </body>
</html>
```

- Change the code of application/controllers/test.php file as shown in the below:

Loading the View:

- The view can be loaded by the **syntax**: `$this->load->view('name');` where, name is the view file, which is being rendered.
- If we have planned to store the view file in some directory then we can use the **syntax**: `$this->load->view('directory-name/name');`
- It is not necessary to specify the extension as php, unless something other than .php is used.
- The index() method is calling the view method and passing the "test" as argument to view() method because we have stored the html coding in "test1.php" file under application/views/test.php.

```
<?php
   class Test1 extends CI_Controller {
      public function index() {
         $this->load->view('test');
      }
   }
?>
```

Output:

```
This example shows how to execute View in CodeIgniter!
```

- Fig. 5.7 shows how everything i.e., MVC works in CodeIgniter.

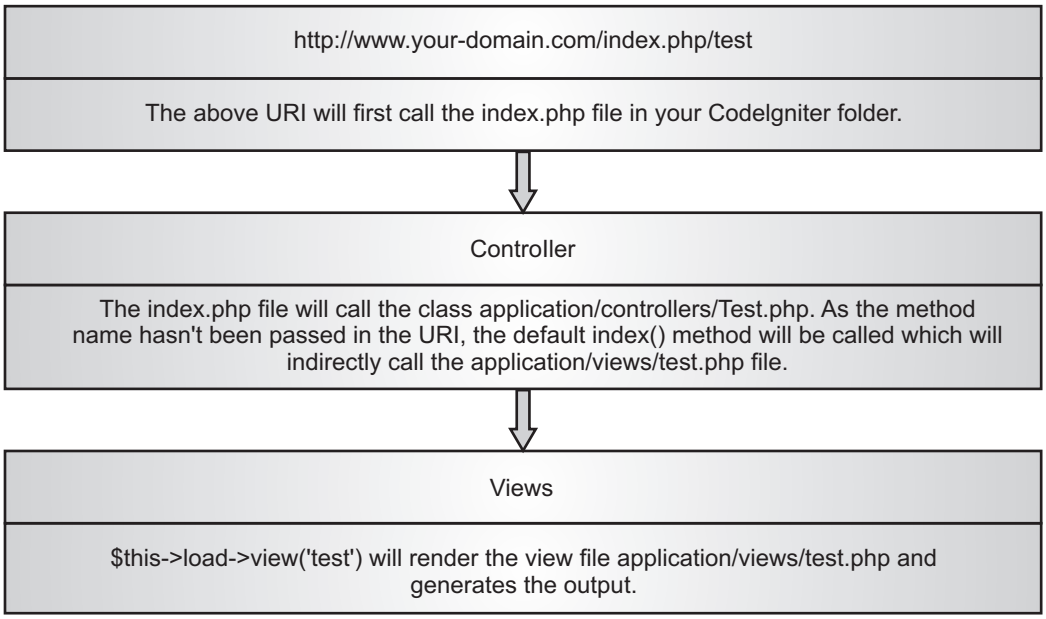


Fig. 5.7

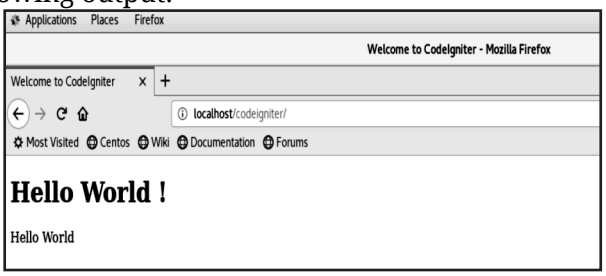
- In the following example we create welcome_message.php is the view file under /application/views/ welcome_message.php folder.

```
<html lang="en">
   <head>
      <title>Welcome to CodeIgniter</title>
   </head>
   <body>
      <h1>Hello World !</h1>
      <div id="body">
      <p>Hello World</p>
      </div>
   </body>
</html>
```

- Welcome.php is the controller file under /application/controller/Welcome.php folder,

```
<?php
   if (!defined('BASEPATH'))
     exit('No direct script access allowed');
     class Welcome extends CI_Controller
     {
        public function index() // default Method
        {
           $this->load->;view('welcome_message'); // load the
                                                  welcome_message.php view
        }
     }
```

- Test the CodeIgniter application using this URL http://localhost/CodeIgnitor. Displays following output:



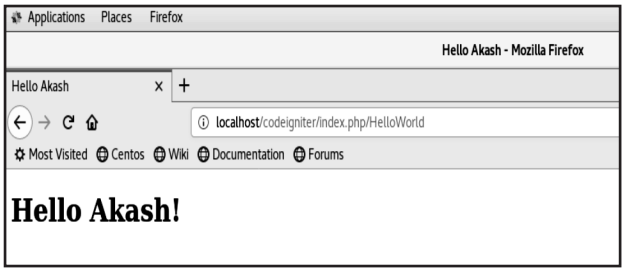
- Controller program HelloWorld.php:

```
<?php
   class HelloWorld extends CI_Controller
   {
      public function index() // default Method
      {
         $this->load->view('helloworld'); // load the helloworld.php view
      }
   }
?>
```

- View program helloworld.php:

```
<!DOCTYPE html>
   <html lang="en">
      <head>
         <meta charset="utf-8">
         <title>Hello Akash</title>
      </head>
      <body>
         <h1>Hello Akash!</h1>
      </body>
</html>
```

Output:



5.6 CODEIGNITER LIBRARIES

- CodeIgniter provide a rich set of libraries. Libraries are an essential part of CodeIgniter as it increases the developing speed of an application. It is located in the system/library.
- CodeIgniter libraries provides a rich set of libraries, which indirectly increase the speed of developing an application.
- The system library in CodeIgniter is located at system/libraries. All we need to do is to load the library that we want to use.
- The library in CodeIgniter can be loaded as given below:

```
$this->load->library('class name');
```

  where, class name is the name of the library that we want to load. If we want to load multiple libraries, then we can simply pass an array as argument to library() function as given below:

```
$this->load->library(array('email', 'table'));
```

Library Classes:

- The library classes in CodeIgniter are located in system/libraries. Each class has various functions to simplify the developing work.

Creating Libraries:

- CodeIgniter framework has rich set of libraries, which we can find in system/libraries folder but CodeIgniter is not just limited to system libraries, we can create our own libraries too, which can be stored in application/libraries folder.
- We can create libraries in following three ways:
  1. Create new library.
  2. Extend the native library.
  3. Replace the native library.
- While creating new library one should keep in mind, the following things:
  1. The name of the file must start with a capital letter e.g. MyCIlibrary.php.
  2. The class name must start with a capital letter e.g. class MyCIlibrary.
  3. The name of the class and name of the file must match.

Mylibrary.php

```
<?php if(!defined('BASEPATH')) exit('No direct script access allowed');
class MyCIlibrary
{
   public function some_function()
   {
   }
}
/* End of file MyCIlibrary.php */
```

Loading the Custom Library:

- The above library can be loaded by simply executing the following line in our controller.

```
$this->load->library('MyCIlibrary');
```

- The MyCIlibrary is the name of our library and we can write it in lowercase as well as uppercase letters. Use the name of the library without ".php" extension.
- After loading the library, we can also call the function of that class as given below:

```
$this->MyCIlibrary->some_function();
```

Extend the Native Library:

- Sometimes, we may need to add our own functionality to the library provided by CodeIgniter. CodeIgniter provides facility by which we can extend the native library and add our own functions.
- To achieve this, we must extend the class of native library class. For example if we want to extend the Email library then it can be done as given below:

```
class MY_Email extends CI_Email
{
}
```

In the above example, MY_Email class is extending the native library's email class CI_Email. This library can be loaded by the standard way of loading email library. Save the above program code in file My_Email.php.

Replace the Native Library:

- In some situations, we do not want to use the native library the way it works and want to replace it with our own way. This can be done by replacing the native library.
- To achieve this, we just need to give the same class name as it is named in native library.
- For example, if we want to replace the Email class, then use the code as shown below. Save the file name with Email.php and give a class name to CI_Email.

Email.php

```
class CI_Email
{
}
```

5.7 WORKING WITH DATABASES

- A database is an organized collection of data, so that it can be easily accessed and managed.
- Like any other framework, we need to interact with the database very often and CodeIgniter makes this job very simple and easy for us. CodeIgniter provides rich set of functionalities to interact with database.
- The database of the site can be configured in application/config/database.php file. In this section, we will understand how the CRUD (Create, Read, Update, Delete) functions work with CodeIgniter.
- We will use Student1 table to select, update, delete, and insert the data in Student1 table as given below:

Student1

r_no	int(8)
s_name	varchar(35)

Connecting to a Database:

- We can connect to database in the following two way:
  1. **Automatic Connecting:** Automatic connection in CodeIgniter can be done by using the file application/config/autoload.php. Automatic connection will load the database for each and every page. We just need to add the database library as given below:

```
$autoload['libraries'] = array('database');
```

2. **Manual Connecting:** If we want database connectivity for only some of the pages, then we can go for manual connecting. We can connect to database manually by adding the following line in any class.

```
$this->load->database();
```

Here, we are not passing any argument because everything is set in the database config file application/config/database.php

Inserting a Record:

- To insert a record in the database, the insert() function is used.

Syntax	`insert([$table = ''[, $set = NULL[, $escape = NULL]]])`
Parameters	• $table (string) is the name of table. • $set (array) is an associative array of field/value pairs. • $escape (bool) whether to escape values and identifiers.
Returns	TRUE on success otherwise FALSE on failure.
Return Type	bool

- The example below shows how to insert a record in Student1 table. The $record is an array in which we have set the data and to insert this data to the table Student1, we just need to pass this array to the insert function in the 2nd argument.

```
$record = array(
          'r_no' => ‘1’,
          's_name' => ‘Akash’
          );
$this->db->insert("Student1", $record);
```

Updating a Record:

- To update a record in the database, the update() function is used along with set() and where() functions.
- The set() function will set the data to be updated.

Syntax	`set($key[, $value = ''[, $escape = NULL]])`
Parameters	• $key (mixed) is an array of field/value pairs. • $value (strin) is field value, if $key is a single field. • $escape (bool) specifies whether to escape values and identifiers.
Returns	CI_DB_query_builder instance (method chaining.
Return Type	CI_DB_query_builder

- The where() function will decide which record to update.

Syntax	`where($key[, $value = NULL[, $escape = NULL]])`
Parameters	• $key (mixed) is name of field to compare, or associative array. • $value (mixed), if a single key, compared to this value. • $escape (bool) specifies whether to escape values and identifiers.
Returns	DB_query_builder instance.
Return Type	object

- The update() function will update data in the database.

Syntax	`update([$table = ''[, $set = NULL[, $where = NULL [, $limit = NULL]]]])`
Parameters	• $table (string) is the name of Table. • $set (array) is an associative array of field/value pairs. • $where (string) is the WHERE clause. • $limit (int) is the LIMIT clause.
Returns	TRUE on success, FALSE on failure.
Return Type	bool

```
$rec = array(
          'r_no' => ‘1’,
          's_name' => ‘Amit’
          );
$this->db->set($rec);
$this->db->where("r_no", ‘1’);
$this->db->update("Student1", $rec);
```

Deleting a Record:

- The delete() function is used to delete a record in the database.

Syntax	`delete([$table = ''[, $where = ''[, $limit = NULL [, $reset_data = TRUE]]]])`
Parameters	• $table (mixed) is the table(s) to delete from; string or array. • $where (string) is the WHERE clause. • $limit (int) is the LIMIT clause. • $reset_data (bool) is TRUE to reset the query “write” clause.
Returns	CI_DB_query_builder instance (method chaining) or FALSE on failure.
Return Type	mixed

- We use the following program code to delete a record in the student1 table. The first argument indicates the name of the table to delete record and the second argument decides which record to delete.

```
$this->db->delete("Student1", "r_no = 1");
```

Selecting a Record:

- To select a record in the database, the get() function.

Syntax	`get([$table = ''[, $limit = NULL[, $offset = NULL]]])`
Parameters	• $table (string) is the table to query array. • $limit (int) is the LIMIT clause. • $offset (int) is the OFFSET clause.
Returns	CI_DB_result instance (method chaining).
Return Type	CI_DB_result

- We use the following program code to get all the records from the database. The first statement fetches all the records from "Student1" table and returns the object, which will be stored in $query object.
- The second statement calls the result() function with $query object to get all the records as array.

```
$qry = $this->db->get("student1");
$data['records'] = $qry->result();
```

Closing a Connection:

- Database connection can be closed manually, by executing the code given below:

```
$this->db->close();
```

- **Example:** In this example we create a controller class called Student_controller.php and save it at application/controller/Student_controller.php
- Following code shows all of the above-mentioned operations. Before executing the following example, we first create a database and table as instructed at the starting and make necessary changes in the database config file stored at application/ config/database.php.

```
<?php
   class Student_controller extends CI_Controller {
      function __construct() {
         parent::__construct();
         $this->load->helper('url');
         $this->load->database();
      }
```

```
public function index() {
   $query = $this->db->get("Student1");
   $data['records'] = $query->result();
   $this->load->helper('url');
   $this->load->view(' Student_view',$data);
}
public function add_student_view() {
   $this->load->helper('form');
   $this->load->view('Stud_add');
}
public function add_student() {
   $this->load->model('Student_Model');
   $data = array(
      'r_no' => $this->input->post('r_no'),
      's_name' => $this->input->post('s_name')
   );
   $this->'Student_Model->insert($data);
   $query = $this->db->get("student1");
   $data['records'] = $query->result();
   $this->load->view('Student_view',$data);
}
public function update_student_view() {
   $this->load->helper('form');
   $r_no = $this->uri->segment('3');
   $query = $this->db->get_where("student1",array("r_no"=>$r_no));
   $data['records'] = $query->result();
   $data['old_r_no'] = $r_no;
   $this->load->view('Stud_edit',$data);
}
public function update_student(){
   $this->load->model('Student_Model');
   $data = array(
      'r_no' => $this->input->post('r_no'),
      's_name' => $this->input->post('s_name')
   );
   $old_r_no = $this->input->post('old_r_no');
```

```
            $this->Student_Model->update($data,$old_r_no);
            $query = $this->db->get("student1");
            $data['records'] = $query->result();
            $this->load->view('Student_view',$data);
         }
         public function delete_student() {
            $this->load->model('Student_Model');
            $r_no = $this->uri->segment('3');
            $this->Student_Model->delete($r_no);
            $query = $this->db->get("Student1");
            $data['records'] = $query->result();
            $this->load->view('Student_view',$data);
         }
      }
   ?>
```

- We create a model class called Stud_Model.php and save it in application/ models/Student_Model.php.

```
<?php
   class Student_Model extends CI_Model {
      function __construct() {
         parent::__construct();
      }
      public function insert($data) {
         if ($this->db->insert("Student1", $data)) {
            return true;
         }
      }
      public function delete($r_no) {
         if ($this->db->delete("Student1", "r_no = ".$r_no)) {
            return true;
         }
      }
      public function update($data,$old_r_no) {
         $this->db->set($data);
         $this->db->where("r_no", $old_r_no);
         $this->db->update("Student1", $data);
      }
   }
?>
```

- We create a view file called Stud_add.php and save it in application/views/Stud_add.php.

```
<!DOCTYPE html>
<html lang = "en">
   <head>
      <meta charset = "utf-8">
      <title>Students Example</title>
   </head>
   <body>
         <?php
            echo form_open('Student_controller/add_student');
            echo form_label('Roll No.');
            echo form_input(array('id'=>'r_no','s_name'=>'r_no'));
            echo "<br/>";
            echo form_label('Name');
            echo form_input(array('id'=>'s_name','name'=>'s_name'));
            echo "<br/>";
            echo form_submit(array('id'=>'submit','value'=>'Add'));
            echo form_close();
         ?>
   </body>
</html>
```

- We create a view file called Stud_edit.php and save it in application/views/Stud_edit.php.

```
<!DOCTYPE html>
<html lang = "en">
   <head>
      <meta charset = "utf-8">
      <title>Students Example</title>
   </head>
   <body>
      <form method = "" action = "">
         <?php
            echo form_open('Stud_controller/update_student');
            echo form_hidden('old_r_no',$old_r_no);
            echo form_label('Roll No.');
```

```
            echo form_input(array('id'⇒r_no',
               'name'⇒r_no','value'⇒$records[0]→r_no));
            echo "
            ";

            echo form_label('Name');
            echo form_input(array('id'⇒s_name','name'⇒s_name',
                                    'value'⇒$records[0]→s_name));
            echo "";
            echo form_submit(array('id'⇒sub mit','value'⇒Edit'));
            echo form_close();
         ?>
      </form>
   </body>
</html>
```

- We create a view file called Stud_view.php and save it in application/views/Stud_view.php

```
<!DOCTYPE html>
<html lang = "en">
   <head>
      <meta charset = "utf-8">
      <title>Students Example</title>
   </head>
   <body>
      <a href = "<?php echo base_url(); ?>
         index.php/Student1/add_view">Add</a>
      <table border = "1">
         <?php
            $i = 1;
            echo "<tr>";
            echo "<td>Sr No.</td>";
            echo "<td>Stud Roll No.</td>";
            echo "<td>Stud Name</td>";
            echo "<td>Edit</td>";
            echo "<td>Delete</td>";
            echo "<tr>";
```

```
            foreach($records as $r) {
               echo "<tr>";
               echo "<td>".$i++."</td>";
               echo "<td>".$r->r_no."</td>";
               echo "<td>".$r->s_name."</td>";
               echo "<td><a href = '".base_url()."index.php/Student1/edit/"
                  .$r->r_no."'>Edit</a></td>";
               echo "<td><a href =
                                 '".base_url()."index.php/Student1/delete/"
               .$r->r_no."'>Delete</a></td>";
               echo "<tr>";
            }
         ?>
      </table>
   </body>
</html>
```

- Add the following line at the end of file routes.php, fill at application/config/routes.php

```
$route['Student1'] = "Student_controller";
$route['Student1/add'] = 'Student_controller/add_student';
$route['Student1/add_view'] = 'Stud_controller/add_student_view';
$route['Student1/edit/(\d+)'] = 'Stud_controller/update_student_view/$1';
$route['Student1/delete/(\d+)'] = 'Student_controller/delete_student/$1';
```

- Now, let us execute this example by visiting the following URL in the browser. Replace the yourssite.com with the URL.

```
http://usersite.com/index.php/student1
```

5.8 ADDING/LOADING JS AND CSS

- Adding JS (JavaScript) and CSS (Cascading Style Sheet) file in CodeIgniter is very simple and easy.
- We have to create JS and CSS folder in root directory and copy all the .js files in JS folder and .css files in CSS folder.
- Example let us assume, we have created one JavaScript file sample.js and one CSS file style.css. Now, to add these files into our views, load URL helper in the controller as follows:

```
$this->load->helper('url');
```

- After loading the URL helper in controller, simply add the below given lines in the view file, to load the sample.js and style.css file in the view as follows:

```
<link rel = "stylesheet" type = "text/css"
                            href = "<?php echo base_url(); ?>css/style.css">
<script type = 'text/javascript' src = "<?php echo
                                        base_url(); ?>js/sample.js"></script>
```

- In following example we create a controller called Test.php and save it in application/ controller/Test.php.

```
<?php
   class Test extends CI_Controller
   {
      public function index()
      {
         $this->load->helper('url');
         $this->load->view('test');
      }
   }
?>
```

- We create a view file called test.php and save it at application/views/test.php.

```
<html lang = "en">
   <head>
      <meta charset = "utf-8">
      <title>CodeIgniter View Example</title>
      <link rel = "stylesheet" type = "text/css"
                            href = "<?php echo base_url(); ?>css/style.css">
      <script type = 'text/javascript' src = "<?php echo
                                         base_url();?>js/sample.js"></script>
   </head>
   <body>
      <a href = 'javascript:test()'>Click Here</a> Execute
                                                        JavaScript Function.
   </body>
</html>
```

- We create a CSS file called style.css and save it at css/style.css.

```
body
{
   background:#FFF;
   color:#000;
}
```

- We create a JS file called sample.js and save it at js/sample.js.

```
function test()
{
   alert('test the function');
}
```

- Then we change the routes.php file in application/config/routes.php to add route for the above controller and add the following line at the end of the file.

```
$route['profiler'] = "Profiler_controller";
$route['profiler/disable'] = "Profiler_controller/disable"
```

- We can use the following URL in the browser to execute the above example:

```
http://yoursite.com/index.php/test
```

5.9 PAGE REDIRECTING

- While building Web application, sometimes we need to redirect a Web page to another Web page.
- CodeIgniter uses the redirect() function for page redirection.

Syntax	`redirect($uri = '', $method = 'auto', $code = NULL).`
Parameters	• $uri (string) is the URI string. • $method (string) redirect method ('auto', 'location' or 'refresh'). • $code (string) is the HTTP Response code (usually 302 or 303).
Return type	void

  - The first argument of redirect() function can have two types of URI. We can pass full site URL or URI segments to the controller we want to direct.
  - The second optional parameter can have any of the three values from auto, location or refresh. The default is auto.
  - The third optional parameter is only available with location redirects and it allows us to send specific HTTP response code.
- In following example we can create a controller called Redirect_controller.php and save it in application/controller/Redirect_controller.php.

```
<?php
   class RedirectCI_controller extends CI_Controller {
      public function index() {
         /*Load the URL helper*/
         $this->load->helper('url');
```

```
            /*Redirect the user to some site*/
            redirect('http://www.niralibooks.com');
        }
        public function computer_graphics() {
            /*Load the URL helper*/
            $this->load->helper('url');
           redirect('http://www.niralibooks.com/computer_graphics/index.htm');
        }
        public function version2() {
            /*Load the URL helper*/
            $this->load->helper('url');
            /*Redirect the user to some internal controller's method*/
            redirect('redirect/computer_graphics');
        }
    }
?>
```

- Then we change the routes.php file in application/config/routes.php to add route for the above controller and add the following line at the end of the file.

```
$route['redirect'] = 'RedirectCI_controller';
$route['redirect/version2'] = 'RedirectCI_controller/version2';
$route['redirect/computer_graphics'] =
                                'RedirectCI_controller/computer_graphics';
```

- We type the following URL in the browser, to execute the example:
  `http://yoursite.com/index.php/redirect`
- The above URL will redirect us to the niralibooks.com website and if we visit the following URL, then it will redirect us to the computer graphics tutorial at niralibooks.com.

```
http://yoursite.com/index.php/redirect/computer_graphics
```

5.10 LOADING DYNAMIC DATA ON PAGE AND SESSION MANAGEMENT

- When building Websites, we often need to track user's activity or action and state and for this purpose, we have to use session. For this purpose CodeIgniter has session class.

Initializing a Session:

- Sessions data are available globally through the Website but to use those data we first need to initialize the session.

- We can do that by executing the following line in constructor:

```
$this->load->library('session');
```

- After loading the session library, we can simply use the session object as given below:

```
$this->session
```

Adding Session Data:

- In PHP, we simply use $_SESSION array to set any data in session as given below.

```
$_SESSION['key'] = value;
```

where 'key' is the key of array and value is assigned on right side of equal to sign.

- The same thing can be done in CodeIgniter as given below:

```
$this->session->set_userdata('any_name', 'any_value');
```

- The set_userdata() function takes two arguments. The first argument, any_name, is the name of the session variable, under which, any_value will be stored.
- The set_userdata() function also supports another **syntax** in which we can pass array to store values as given below:

```
$data = array(
       'username'  => 'Amar',
        'email'     => Amar12345@gamil.com',
        'logged_in' => TRUE
       );
$this->session->set_userdata($data);
```

Removing Session Data:

- In PHP, we can remove data stored in session using the unset() function. The **syntax** is given below:

```
unset($_SESSION['any_name']);
```

- Removing session data in CodeIgniter is very simple and easy. The following version of unset_userdata() function will remove only one variable from session.

```
$this->session->unset_userdata('any_name');
```

- If we want to remove more values from session or to remove an entire array we can use the below version of unset_userdata() function.

```
$this->session->unset_userdata($array_elements);
```

Fetching Session Data:

- After setting data in session, we can also retrieve that data for this purpose use the **Userdata()** function. This function will return NULL if the data we are trying to access is not available.

```
$name1 = $this->session->userdata('name');
```

- In the following example we create a controller class called Session_controller.php and save it in application/ controller/Session_controller.php.

```
<?php
   class SessionCI_controller1 extends CI_Controller {
      public function index() {
         //loading session library
         $this->load->library('session');
         //adding data to session
         $this->session->set_userdata('s_name','Amit');
         $this->load->view('session_view');
      }
      public function unset_session_data() {
         //loading session library
         $this->load->library('session');
         //removing session data
         $this->session->unset_userdata('s_name');
         $this->load->view('session_view');
      }
   }
?>
```

- Then we create a view file called session_view1.php and save it in application/ views/session_view1.php.

```
<!DOCTYPE html>
<html lang = "en">
  <head>
     <meta charset = "utf-8">
     <title>This is CodeIgniter Session handling Example</title>
  </head>
   <body>
      Welcome <?php echo $this->session->userdata('s_name'); ?>
      <br>
      <a href = 'http://localhost:80/
                           CodeIgniter3.0.1/index.php/sessionex/unset'>
         Click this</a> to unset session data.
   </body>
</html>
```

- Then we make the changes in the routes.php file in application/config/routes.php and add the following line at the end of the file.

```
$route['sessionex'] = 'SessionCI_Controller1';
```

5.11 COOKIES MANAGEMENT

- Cookie is a small piece of data sent from Web server to store on client's/browser's computer.
- CodeIgniter has one helper called "Cookie Helper" for cookie management. The Cookie Helper contains functions that assist in working with cookies.

Syntax	`set_cookie($name[, $value = ''[, $expire = ''`
	`[, $domain = ''[, $path = '/'[, $prefix = ''`
	`[, $secure = FALSE[, $httponly = FALSE]]]]]]])`

Parameters	• $name (mixed) is the cookie name or associative array of all of the parameters available to this function. • $value (string) is the cookie value. • $expire (int) is the number of seconds until expiration. • $domain (string) is the cookie domain (usually: .yourdomain.com). • $path (string) is the cookie path. • $prefix (string) is the cookie name prefix. • $secure (bool) specifies whether to only send the cookie through HTTPs. • $httponly (bool) specifies whether to hide the cookie from JavaScript.
Return Type	Void

- The set_cookie(), get_cookie() and delete_cookie() are the three functions involved in CodeIgniter cookie.
- The **set_cookie()** function sets the cookie in which all the values can be passed in two ways. In the first way, only array can be passed and in the second way, individual parameters can also be passed.

Syntax	`get_cookie($index[, $xss_clean = NULL]])`
Parameters	• $index (string) is the cookie name. • $xss_clean (bool) specifies whether to apply XSS filtering to the returned value.
Return	The cookie value or NULL if not found.
Return Type	Mixed

- The **get_cookie()** function is used to get the cookie that has been set using the set_cookie() function.

Syntax	`delete_cookie($name[, $domain = ''[, $path = '/' [, $prefix = '']]])`
Parameters	• $name (string) is the cookie name. • $domain (string) is the cookie domain (usually: .yourdomain.com). • $path (string) is the cookie path. • $prefix (string) is the cookie name prefix.
Return Type	void

- The **delete_cookie()** function is used to delete the cookie.
- In the following example we create a controller called CookieCI_controller1.php and save it at application/controller/CookieCI_controller1.php.

```
<?php
   class CookieCI_controller1 extends CI_Controller {
      function __construct() {
         parent::__construct();
         $this->load->helper(array('cookie', 'url'));
      }
      public function index() {
         set_cookie('cookie_name1','cookie_value1','3600');
         $this->load->view('Cookie_view1');
      }
      public function disp_cookie() {
         echo get_cookie('cookie_name1');
         $this->load->view('Cookie_view1');
      }
      public function delcookie() {
         delete_cookie('cookie_name1');
         redirect('cookie/display');
      }
   }
?>
```

- Then we create a view file called CookieCI_view1.php and save it at application/views/CookieCI_view1.php.

```
<html lang = "en">
   <head>
      <meta charset = "utf-8">
      <title>Example of CodeIgniter View</title>
   </head>
   <body>
      <a href = 'display'>Click</a> To view the cookie…<br>
      <a href = 'delete'>Click</a> To delete the cookie…
   </body>
</html>
```

- We change the routes.php file in application/config/routes.php to add route for the above controller and add the following line at the end of the file.

```
$route['cookie'] = "CookieCI_controller1";
$route['cookie/display'] = "CookieCI_controller/disp_cookie";
$route['cookie/delete'] = "CookieCI_controller/delcookie";
```

- After that, we can execute the following URL in the browser to execute the example,

```
http://usersite.com/index.php/cookie
```

Output:

```
Click To view the cookie…
Click To delete the cookie…
```

PRACTICE QUESTIONS

Q. I Multiple Choice Questions:

1. Which is a powerful PHP framework used to create full-featured web applications?
   (a) CodeIgniter (b) PHPIgniter
   (c) JavaIgniter (d) HTMLIgniter
2. CodeIgniter's source code is maintained at,
   (a) HubGit (b) GitHub
   (c) GitIgniter (d) GitCodeIgniter
3. Why CodeIgniter?
   (a) Clear documentation (b) Strong security
   (c) Simple solutions over complexity (d) All of the mentioned
4. Which is used in CodeIgniter for management cookies (a small piece of data sent from web server to store on client's computer)?
   (a) Session Helper (b) Code Helper
   (c) Cookie Helper (d) None of the mentioned

5. Which is an architecture that gives a separate logical view from the presentation view?
   (a) MVC (Model-View-Controller) (b) VMC (View-Model-Controller)
   (c) CVM (Control-View-Mode-Controller) (d) All of the mentioned
6. CodeIgniter file structure is mainly divided into,
   (a) Application (contains all the code of the project we are working on).
   (b) System (all action of CodeIgniter application happens here).
   (c) User_guide (offline CodeIgniter guide helps in which we can learn here all the functions, libraries, helpers of CodeIgniter)
   (d) All of the mentioned
7. CodeIgniter framework is based on,
   (a) OpenMVC (b) MVC
   (c) VCM (d) None of the mentioned
8. Which part of MVC commands the Model to perform the appropriate action as requested by the user?
   (a) Model (b) View
   (c) Controller (d) All of the mentioned
9. CodeIgniter framework is mainly driven by the technology of PHP which contains resource such as,
   (a) plug-ins (b) helpers
   (c) libraries (d) All of the mentioned
10. In MVC which is deals with data presented to the end user?
   (a) Model (b) View
   (c) Controller (d) All of the mentioned
11. Which is of the following is not advantages of MVC architecture?
   (a) Loose coupling (b) Flexibility
   (c) Error reporting (d) Increased productivity
12. Which is following not a way to create CodeIgnitor libraries?
   (a) create new library (b) extend the native library
   (c) replace the native library (d) modify/update library

Answers

1. (a)	2. (b)	3. (d)	4. (c)	5. (a)	6. (d)	7. (b)	8. (c)	9. (d)	10. (b)
11. (c)	12. (d)								

Q. II Fill in the Blanks:

1. CodeIgniter is a high performance _____ framework for developing MVC-based web applications.
2. _____ is an open-source software rapid development web framework, for use in building dynamic web sites with PHP.
3. Sessions data are available globally through the site but to use those data we _____ need to initialize the session.
4. _____ is a web based hosting service, offers distributed version control and source code management functionality.
5. The _____ is responsible for interacting with data sources. In other words, model usually writes data into the database, provides a mechanism for editing and updating, and deleting data.
6. _____ is the information that is presented in front of users.
7. The _____ serves as an intermediary between the Model, the View and any other resources needed to process the HTTP request and generate a Web page.
8. In CodeIgniter, go to application/config/databse.php for _____ configuration file.
9. The Cookie _____ contains functions that assist in working with cookies.
10. _____ session data in CodeIgniterwe can use unset_userdata() function.
11. CodeIgniter gives access to its session data through session handlers' mechanism provided by PHP. Using session data is as simple as manipulating (read, set and unset values) the _____ array.

Answers

1. PHP	2. CodeIgniter	3. first	4. GitHub
5. Model	6. View	7. Controller	8. database
9. Helper	10. Removing	11. $_SESSION	

Q. III State True or False:

1. CodeIgniter contains libraries, simple interface and logical structure to access these libraries, plug-ins, helpers and some other resources which solve the complex functions of PHP more easily maintaining a high performance.
2. Adding JavaScript and CSS (Cascading Style Sheet) file in CodeIgniter is very simple. We have to create JS and CSS folder in root directory and copy all the .js files in JS folder and .css files in CSS folder.
3. CodeIgniter uses MVC (Model-View-Controller) architecture.
4. CodeIgniter source code is maintained at GitHub.

5. In MVC, the Model represents the data structure or resources, View represent the information that is being presented to a user and Controller is the intermediary between models and view to process HTTP request and generates a Web page.
6. CodeIgniter is Python driven framework.
7. The database of the site can be configured in application/config/database.php file.
8. The get_cookie() function is used to get the cookie that has been set using the set_cookie() function.
9. View and model can be called by controller.
10. While building web application, we often need to redirect the user from one page to another page. CodeIgniter redirect() function is used for redirection purpose.
11. Validation ensures that the data that we are getting is proper and valid to store or process in the Web application.
12. Session data in CodeIgniter is simply an array associated with a particular session ID (cookie).

Answers

1. (T)	2. (T)	3. (T)	4. (T)	5. (T)	6. (F)	7. (T)	8. (T)	9. (T)	10. (T)
11. (T)	12. (T)								

Q. IV Answer the following Questions:

(A) Short Answer Questions:

1. What is the purpose of CodeIgniter framework?
2. List features of CodeIgniter.
3. Define is MVC?
4. “CodeIgniter is PHP driven framework” Comment this sentence.
5. Which three components are the pillars for CodeIgniter application architecture?
6. What is meant by CodeIgniter library.
7. What is helper?
8. What is cookie?
9. Define page redirection.

(B) Long Answer Questions:

1. What is CodeIgniter? Give steps for installation of CodeIgniter.
2. Enlist features of CodeIgniter.
3. With the help of diagram describe CodeIgniter application architecture.
4. What is MVC architecture? Explain diagrammatically.
5. Write shot note on: Basic concept of CodeIgniter.
6. What are CodeIgniter libraries? Explain in detail.
7. How to load external JS and CSS in a Web page? Explain with example.

8. What is redirection? How to redirect a Web page to another? Explain with example.
9. What is cookie? List functions for handing cookies in CodeIgniter.
10. What is session? Which functions are used to initializing and destroying a session in CodeIgniter?
11. How to redirecting controllers? Explain in detail.
12. Write short note on: Working of databases.

■■■